IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

ADVANCED CODING TECHNOLOGIES LLC, Plaintiff, v. LG ELECTRONICS INC. and LG ELECTRONICS U.S.A., INC., Defendants.	Case No. 2:22-cv-00501-JRG (LEAD CASE) **JURY TRIAL DEMANDED**
ADVANCED CODING TECHNOLOGIES LLC, Plaintiff, v. SAMSUNG ELECTRONICS CO., LTD. and SAMSUNG ELECTRONICS AMERICA INC., Defendants.	Case No. 2:22-cv-00499-JRG (MEMBER CASE) **JURY TRIAL DEMANDED**

PLAINTIFF'S DISCLOSURE OF ASSERTED CLAIMS
AND INFRINGEMENT CONTENTIONS

Plaintiff Advanced Coding Technologies LLC ("ACT" or "Plaintiff") hereby makes the following infringement disclosures under the Local Patent Rules with respect to United States Patent Nos. 6,845,128 (the "'128 Patent"), 8,090,025 (the "'025 Patent"), 8,139,150 (the "'150 Patent"), 9,445,041 (the "'041 Patent"), 9,986,303 (the "'303 Patent"), and 10,218,995 (the "'995 Patent") (collectively, the "ACT Patents" or "Asserted Patents"). ACT's investigation is ongoing and discovery is in its preliminary stages. Accordingly, these disclosures are based on information

available to ACT at this time. ACT reserves the right to supplement this disclosure after further discovery from Defendants and non-parties, particularly documents and other discovery regarding Defendants' accused devices. ACT also reserves the right to assert additional claims of the ACT Patents, accuse different products, or find alternative literal and/or equivalent infringing elements in Defendants' products.

I. DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS PURSUANT TO LOCAL PATENT RULE 3-1

A. ASSERTED CLAIMS

Defendants Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. (collectively, "Samsung" or "Defendants) have infringed and/or continue to infringe one or more of the following claims of the ACT Patents in connection with the Accused Products set forth below:

- Claims 1-4 of the '128 Patent;
- Claims 6-8 and 10 of the '025 Patent;
- Claims 1-4 of the '150 Patent;
- Claims 1, 2, 4, and 6 of the '041 Patent;
- Claim 1 of the '303 Patent; and
- Claims 2-4 and 11 of the '995 Patent.

B. ACCUSED INSTRUMENTALITIES

1. Accused Products

(a) '128 Patent

ACT is currently aware that Defendants have and continue to directly infringe the '128 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States

products that satisfy each and every limitation of one or more claims of the ’128 Patent. Such products include each device made, used, offered for sale, sold, and/or imported into the United States containing NVIDIA processors using the “Turing” or “Ampere” architectures including, but not limited to, the Samsung Galaxy Book Odyssey, Samsung Galaxy Book3 Ultra, Samsung Notebook Odyssey and Samsung Notebook 7 Force, which are video-emphasis encoding apparatuses.

(b) ’025, ’303, and ’995 Patents

ACT is currently aware that Defendants have infringed and continue to directly infringe the ’025, ’303, and ’995 Patents, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’025, ’303, and ’995 Patents. Such products include at least Samsung Televisions compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, AU7000, AU8000, AU9000, BU8000, BU8500, Q60A, Q60B, Q60C, Q700T, Q70A, Q70B, Q70C, Q800T, Q80A, Q80B, Q80C, Q900TS, Q90T, Q950TS, Q95T, QN700A, QN700B, QN700C, QN800A, QN800B, QN800C, QN85A, QN85B, QN85C, QN900A, QN900B, QN900C, QN90A, QN90B, QN91A, QN95A, QN95B, QN95C, S90C, S95B, S95C, The Frame 2021, The Frame 2022, the Serif 2022, Q95T, Q90T, Q80T, Q70T, Q60T, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000, LS01T, LS03A, LS03T, LS05T, LST7, QP75A-8K, LSP9T, HBU8000, and HQ60B. Products also include at least Samsung Handsets compliant with compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra. Products also include at least Samsung Laptops compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, Galaxy Book Pro, Galaxy Book Pro 360, Galaxy Book2

Pro, Galaxy Book2 Pro 360, Galaxy Book2 Business 14', Galaxy Book Odyssey, Galaxy Book3 Ultra, Galaxy Book3 Pro, Galaxy Book3 Pro 360, and Galaxy Book3 360.

(c) '150 Patent

ACT is currently aware that Defendants have infringed and continue to directly infringe the '150 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the '150 Patent. Such products include at least Televisions and Handsets compliant with the H.264 Multi-View Coding (MVC) Standard including, but not limited to, DBJ Series, QN900A, QN800A, QN700A, Q950T, Q900T, Q800T, Q700T, QN95A, QN91A, QN90A, QN85A, Q80A, Q70A, Q95T, Q90T, Q80T, Q70T, Q60T, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, and TU7000.

(d) '041 Patent

ACT is currently aware that Defendants have infringed and continue to directly infringe the '041 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the '041 Patent. Such products include at least Samsung's Handsets including, but not limited to, the Galaxy A13 5G, Galaxy A22 5G, Galaxy A32 5G, Galaxy A33 5G, Galaxy A40, Galaxy A42 5G, Galaxy A50, Galaxy A50s, Galaxy A51 5G, Galaxy A52 5G, Galaxy A52s 5G, Galaxy A53 5G, Galaxy A70, Galaxy A70s, Galaxy A71 5G, Galaxy A73 5G, Galaxy A80, Galaxy A90 5G, Galaxy S9, Galaxy S9+, Galaxy Note9, Galaxy S10e, Galaxy S10, Galaxy S10+, Galaxy S10 Lite, Galaxy Fold 5G, Galaxy Note 10 5G, Galaxy Note 10+ 5G, Galaxy Note 20 5G, Galaxy Note 20 Ultra

5G, Galaxy Quantum 2, Galaxy S10 5G, Galaxy S20 5G, Galaxy S20 FE 5G, Galaxy S20 Ultra 5G, Galaxy S20 UW, Galaxy S20+ 5G, Galaxy S21 5G, Galaxy S21 Ultra 5G, Galaxy S21+ 5G, Galaxy S22, Galaxy S22 5G, Galaxy S22 Ultra, Galaxy S22+, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra, Galaxy Tab S7 5G, Galaxy Tab S7 FE 5G, Galaxy Tab S7+ 5G, Galaxy Tab S8+ 5G, Galaxy Z Flip 5G, Galaxy Z Flip3 5G, Galaxy Z Flip4 5G, Galaxy Z Fold 2 5G, the Galaxy Z Fold3 5G, and Galaxy Z Fold4 5G.

2. Claim Charts

Claim charts identifying a location of every element of every asserted claim of the ACT Patents within the Accused Products are attached hereto as Appendices A-F. ACT believes that the Accused Products cited in the claim charts are representative of all versions of the Accused Products. The Accused Products include the hardware with associated software and the charts also reference exemplary code to the extent available.

ACT reserves the right to amend these claim charts, as well as other information contained in this document and the Appendices attached hereto, to incorporate new information learned during the course of discovery including, but not limited to, information that is not publicly available or readily discernible without discovery. ACT further reserves the right to amend these claim charts, as well as other information contained in this document and the exhibits attached hereto, pursuant to Local Patent Rules 3-1(g) and 3-6.

C. LITERAL INFRINGEMENT AND DOCTRINE OF EQUIVALENTS

ACT asserts that, under the proper construction of the asserted claims and their claim terms, the limitations of the asserted claims of the ACT Patents are literally present in the Accused Products as set forth in the claim charts attached hereto as Appendices A–F.

ACT contends that any and all elements found not to be literally infringed are infringed under the doctrine of equivalents because the differences between the claimed inventions and the Accused Products, if any, are insubstantial.

ACT contends that Defendants directly infringe the asserted claims by making, using, offering for sale, selling, and importing into the United States the Accused Products. ACT also contends that Defendants indirectly infringe by contributing to and/or inducing others (*e.g.*, Defendants' service providers or partners, suppliers, Defendants' customers or its customers' customers) to directly infringe those claims by making or using the Accused Products. In addition, ACT contends that Defendants directly infringe to the extent that activities of third parties may be legally attributable to Defendants.

ACT contends that Defendants induce infringement of the ACT Patents by others (manufacturers and distributors Accused Products, end-users of Accused Products, and Defendants' customers) by inducing others to make or use the Accused Products with the knowledge that making or using directly infringes the ACT Patents or, in the alternative, with the belief that there was a high probability that others infringe the ACT Patents, but while remaining willfully blind to the infringement.

ACT also contends that Defendants contributorily infringe with the knowledge that the Accused Products, or the use thereof, infringe the Asserted Patents. Defendants knowingly and intentionally contributed to the direct infringement of the Asserted Patents by others, by supplying Accused Products that embody a material part of the claimed invention of the Asserted Patents that are known by Defendants to be specially made or adapted for use in an infringing manner and are not staple articles with substantial non-infringing uses.

Pursuant to Local Patent Rule 3-6(a)(1), ACT reserves the right to amend its Infringement Contentions as to literal infringement or infringement under the doctrine of equivalents in light of the Court's claim construction.

D. PRIORITY DATES

Each asserted claim of the '128 Patent is entitled to at least the priority date of its earliest application, U.S. Provisional Patent Application No. 10/067,815, with a filing date of February 8, 2002. Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application Nos. 2001-0033401 and 2001-033403, filed February 9, 2001.

Each asserted claim of the '025 Patent is entitled to at least the filing date of its earliest application, U.S. Provisional Patent Application No. 11/787,623, filed April 17, 2007. Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application No. 2006-112995, filed April 17, 2006.

Each asserted claim of the '150 Patent is entitled to at least the filing date of its earliest application, U.S. Provisional Patent Application No. 11/907,188, filed October 10, 2007. Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application No. 2006-279364, filed October 13, 2006.

Each asserted claim of the '041 Patent is entitled to at least the filing date of its earliest application, U.S. Provisional Patent Application No. 14/141,238, filed December 26, 2013. Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application No. 2012-288402, filed December 28, 2012.

Each asserted claim of the '303 Patent is entitled to at least the priority date of its earliest application, U.S. Patent Application No. 14/657,359, with a filing date of March 13, 2015.

Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application No. 2014-071975, filed March 31, 2014.

Each asserted claim of the '995 Patent is entitled to at least the filing date of its earliest application, U.S. Divisional Patent Application No. 12/995,039, filed May 28, 2009. Furthermore, each asserted claim is entitled to at least the filing date of its earliest foreign application, Japan Patent Application No. P2008-142433, filed May 30, 2008.

ACT makes this disclosure without prejudice to its right to prove an earlier date of invention.

II. PRODUCTION OF DOCUMENTS PURSUANT TO LOCAL PATENT RULE 3-2

ACT is producing or making available for inspection documents that are in ACT's possession, custody, or control as set forth in Local Patent Rule 3-2. A Production Index identifying these documents is attached hereto.

This preliminary identification of documents is for convenience and is not an admission that each document falls within any exemplary categories in the Local Patent Rules, or that any document qualifies as prior art. Thus, ACT reserves its right to add to, delete from, or otherwise modify its disclosures in this section as its investigation proceeds.

Dated: May 16, 2023

Respectfully submitted,

/s/ Alfred R. Fabricant

Alfred R. Fabricant
NY Bar No. 2219392
Email: ffabricant@fabricantllp.com
Peter Lambrianakos
NY Bar No. 2894392
Email: plambrianakos@fabricantllp.com
Vincent J. Rubino, III
NY Bar No. 4557435
Email: vrubino@fabricantllp.com
Joseph M. Mercadante

NY Bar No. 4784930
Email: jmercadante@fabricantllp.com
FABRICANT LLP
411 Theodore Fremd Avenue
Suite 206 South
Rye, New York 10580
Telephone: (212) 257-5797
Facsimile: (212) 257-5796
Justin Kurt Truelove
Texas Bar No. 24013653
Email: kurt@truelovelawfirm.com
TRUELOVE LAW FIRM, PLLC
100 West Houston Street
Marshall, Texas 75670
Telephone: (903) 938-8321
Facsimile: (903) 215-8510

ATTORNEYS FOR PLAINTIFF ADVANCED CODING TECHNOLOGIES, LLC

CERTIFICATE OF SERVICE

I hereby certify that on May 16, 2023, a true and correct copy of the above and foregoing document has been served by email on all counsel of record.

/s/ Alfred R. Fabricant
Alfred R. Fabricant

ACT 3-2 PRODUCTION INDEX

3-2(a)

None.

3-2(b)

Plaintiff is not in possession, custody, or control of any documents responsive to P.R. 3-2(b). Plaintiff states that such documents are in the possession of Victor Company of Japan, Ltd. and/or JVC Kenwood Corporation.

3-2(c)

Bates Start	Bates End
ACT-SAMSUNG00000001	ACT-SAMSUNG00002210

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claims 1-4 (the "Asserted Claims") of U.S. Patent No. 6,845,128 (the "'128 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q, depicted in this chart, is believed to be representative of other laptops including at least the Samsung Galaxy Book Odyssey, Samsung Galaxy Book3 Ultra, Samsung Notebook Odyssey, and Samsung Notebook 7 Force models for the purposes of infringement of the '128 Patent. Based on publicly available information, Advanced Coding Technologies believes that the Samsung Galaxy Book Odyssey, charted herein, and other television/display products share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the listed laptops all purport to comprise NVENC hardware that applies emphasis processing. Should discovery yield documents or information demonstrating that Defendant sells other infringing products which are not represented by the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q, Advanced Coding Technologies reserves its right to amend its infringement contentions to include a chart mapping the claims of the '128 Patent to the structure and function of the newly-discovered products.

Samsung directly infringes each of the Asserted Claims by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

Samsung indirectly infringes the Asserted Claims in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through their operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claim; and (ii) marketing and advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers,

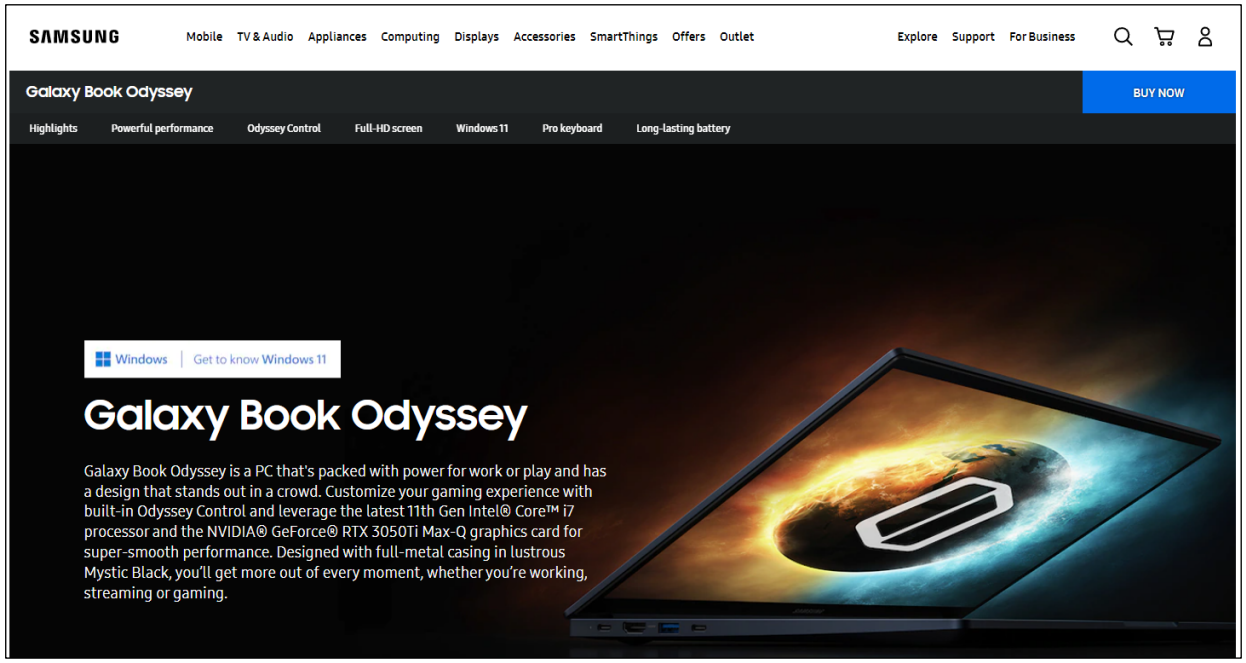
including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claims, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claims but has taken deliberate actions to avoid learning these facts.

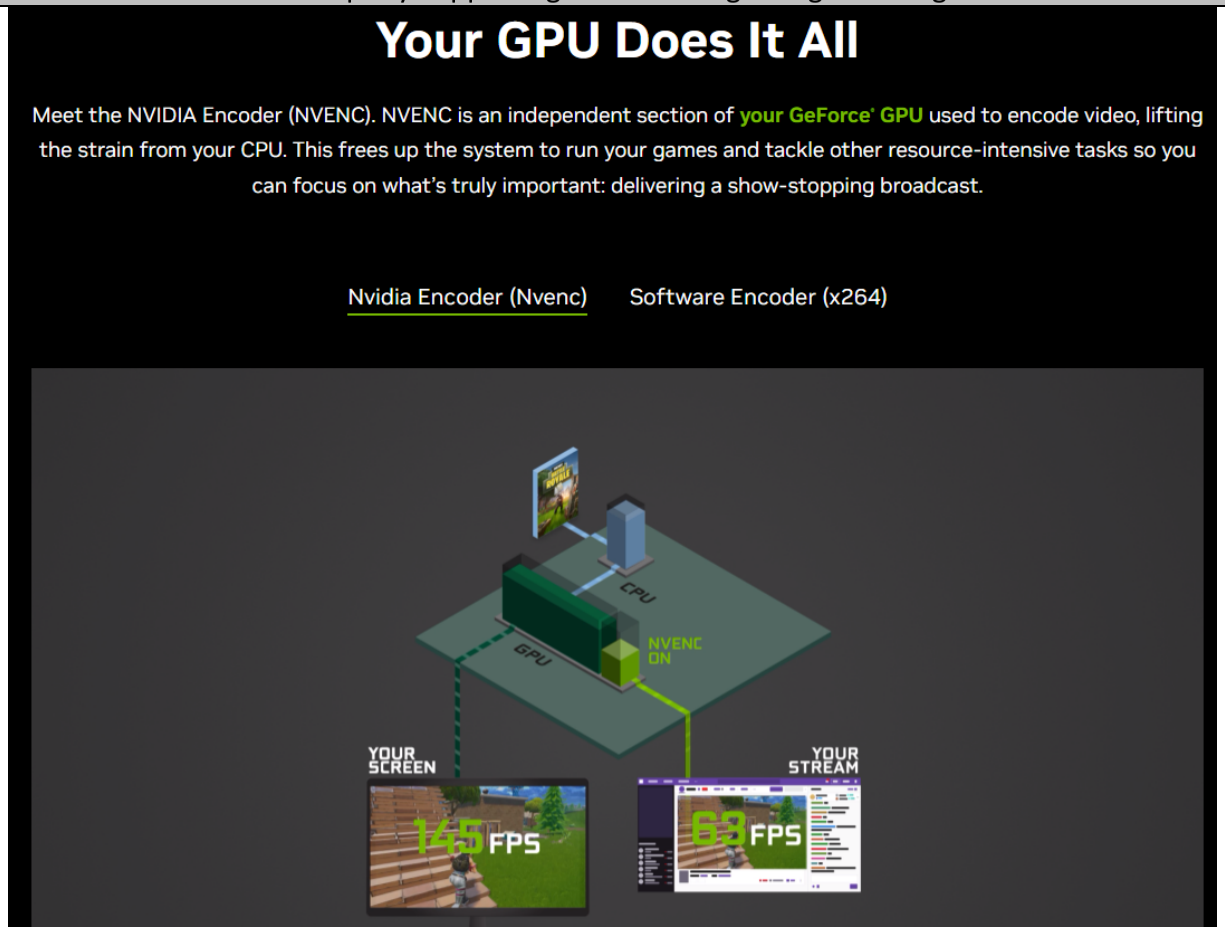
Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

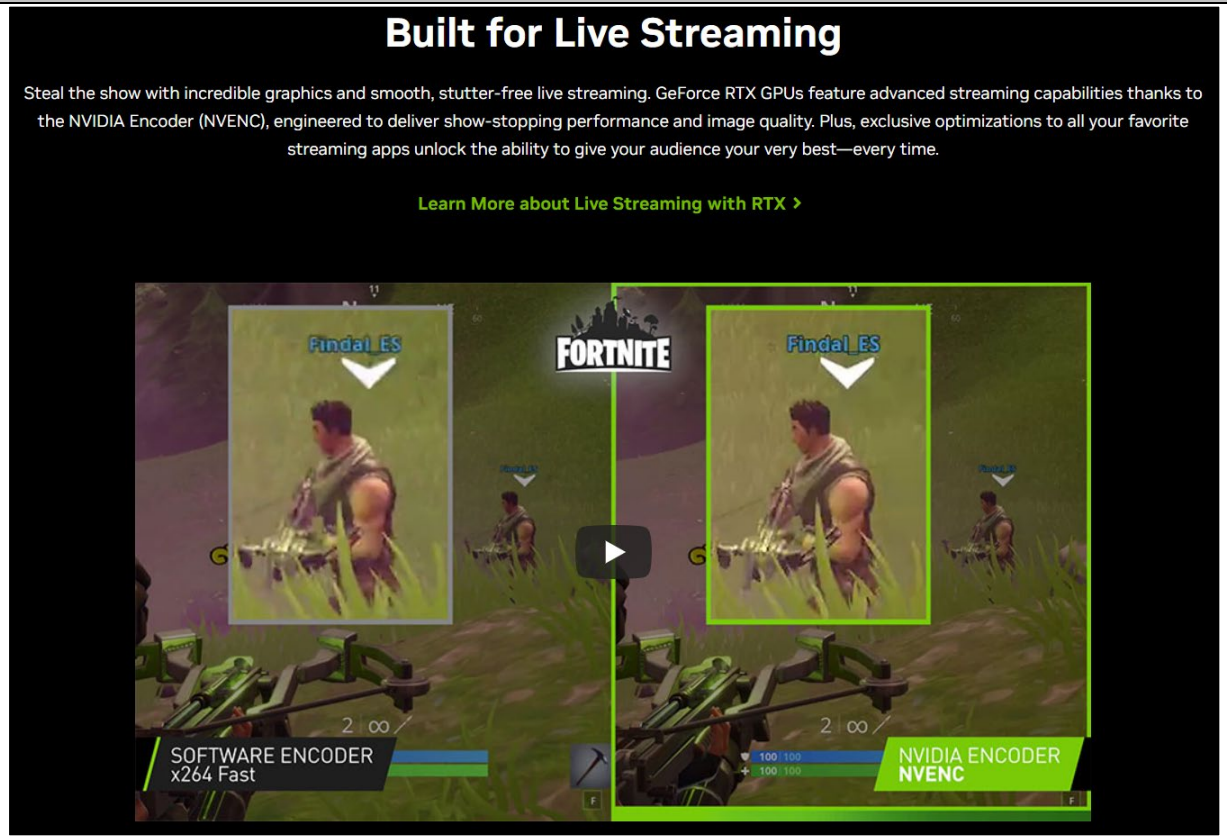
ACT does not concede that any claims of the '128 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

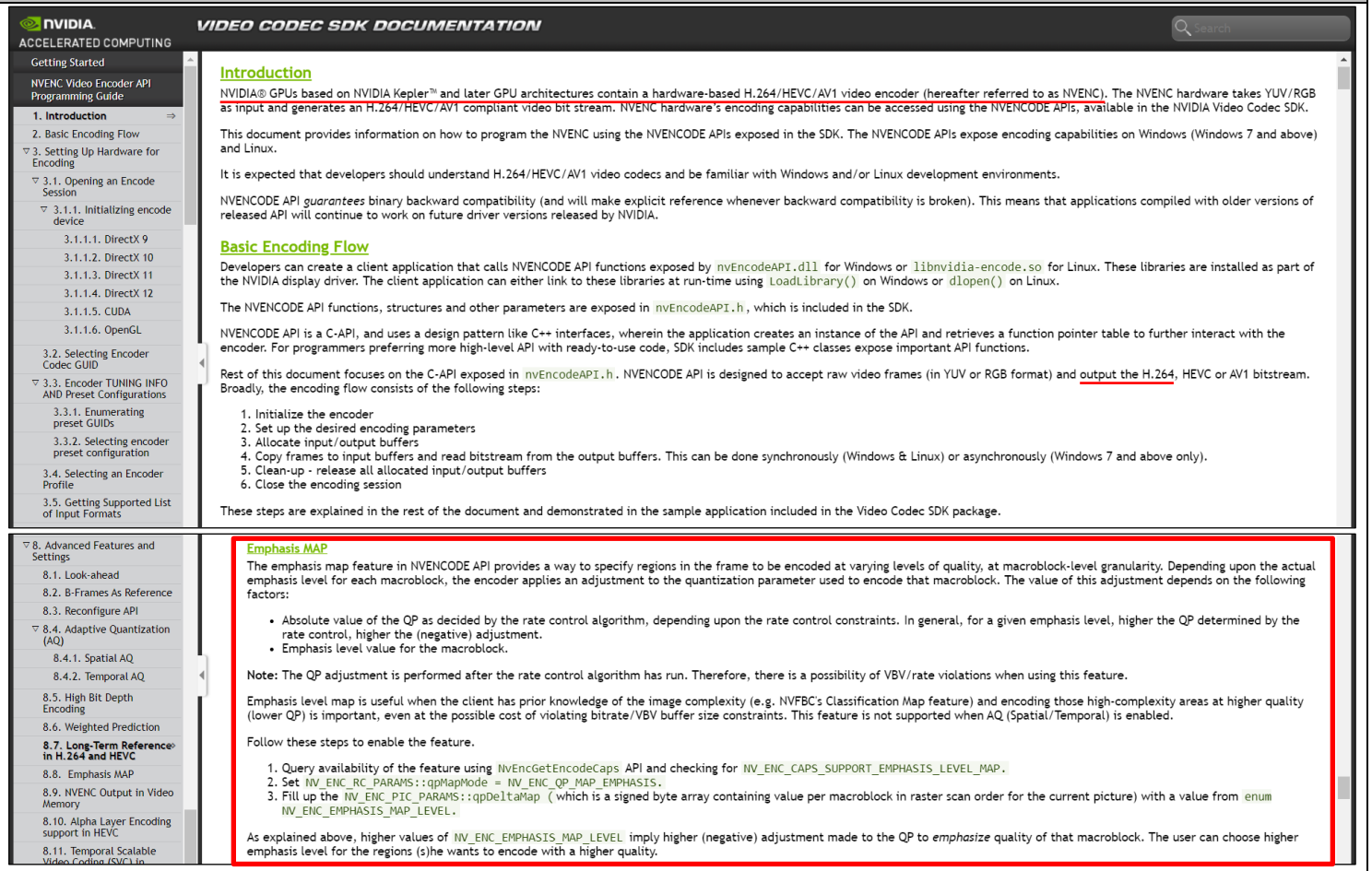
Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1P] A video-emphasis encoding	The Samsung Accused Products comprise: A video-emphasis encoding apparatus for applying an emphasis processing to an input video signal to obtain a video bitstream. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
apparatus for applying an emphasis processing to an input video signal to obtain a video bitstream comprising:	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q is a video-emphasis encoding apparatus for applying an emphasis processing to an input video signal (*i.e.*, for applying an emphasis map to an input video signal using the NVIDIA NVENC encoder) to obtain a video bitstream (*e.g.*, an H.264 video bitstream).  https://www.samsung.com/us/computing/galaxy-book-odyssey/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.nvidia.com/en-us/geforce/broadcasting/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.nvidia.com/en-my/geforce/laptops/30-series/#features

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



NVIDIA ACCELERATED COMPUTING — VIDEO CODEC SDK DOCUMENTATION

Getting Started

NVENC Video Encoder API Programming Guide

1. Introduction
2. Basic Encoding Flow
3. Setting Up Hardware for Encoding
   - 3.1. Opening an Encode Session
     - 3.1.1. Initializing encode device
       - 3.1.1.1. DirectX 9
       - 3.1.1.2. DirectX 10
       - 3.1.1.3. DirectX 11
       - 3.1.1.4. DirectX 12
       - 3.1.1.5. CUDA
       - 3.1.1.6. OpenGL
   - 3.2. Selecting Encoder Codec GUID
   - 3.3. Encoder TUNING INFO AND Preset Configurations
     - 3.3.1. Enumerating preset GUIDs
     - 3.3.2. Selecting encoder preset configuration
   - 3.4. Selecting an Encoder Profile
   - 3.5. Getting Supported List of Input Formats

Introduction

NVIDIA® GPUs based on NVIDIA Kepler™ and later GPU architectures contain a hardware-based H.264/HEVC/AV1 video encoder (hereafter referred to as NVENC). The NVENC hardware takes YUV/RGB as input and generates an H.264/HEVC/AV1 compliant video bit stream. NVENC hardware's encoding capabilities can be accessed using the NVENCODE APIs, available in the NVIDIA Video Codec SDK.

This document provides information on how to program the NVENC using the NVENCODE APIs exposed in the SDK. The NVENCODE APIs expose encoding capabilities on Windows (Windows 7 and above) and Linux.

It is expected that developers should understand H.264/HEVC/AV1 video codecs and be familiar with Windows and/or Linux development environments.

NVENCODE API *guarantees* binary backward compatibility (and will make explicit reference whenever backward compatibility is broken). This means that applications compiled with older versions of released API will continue to work on future driver versions released by NVIDIA.

Basic Encoding Flow

Developers can create a client application that calls NVENCODE API functions exposed by `nvEncodeAPI.dll` for Windows or `libnvidia-encode.so` for Linux. These libraries are installed as part of the NVIDIA display driver. The client application can either link to these libraries at run-time using `LoadLibrary()` on Windows or `dlopen()` on Linux.

The NVENCODE API functions, structures and other parameters are exposed in `nvEncodeAPI.h`, which is included in the SDK.

NVENCODE API is a C-API, and uses a design pattern like C++ interfaces, wherein the application creates an instance of the API and retrieves a function pointer table to further interact with the encoder. For programmers preferring more high-level API with ready-to-use code, SDK includes sample C++ classes expose important API functions.

Rest of this document focuses on the C-API exposed in `nvEncodeAPI.h`. NVENCODE API is designed to accept raw video frames (in YUV or RGB format) and output the H.264, HEVC or AV1 bitstream. Broadly, the encoding flow consists of the following steps:

1. Initialize the encoder
2. Set up the desired encoding parameters
3. Allocate input/output buffers
4. Copy frames to input buffers and read bitstream from the output buffers. This can be done synchronously (Windows & Linux) or asynchronously (Windows 7 and above only).
5. Clean-up - release all allocated input/output buffers
6. Close the encoding session

These steps are explained in the rest of the document and demonstrated in the sample application included in the Video Codec SDK package.

8. Advanced Features and Settings
   - 8.1. Look-ahead
   - 8.2. B-Frames As Reference
   - 8.3. Reconfigure API
   - 8.4. Adaptive Quantization (AQ)
     - 8.4.1. Spatial AQ
     - 8.4.2. Temporal AQ
   - 8.5. High Bit Depth Encoding
   - 8.6. Weighted Prediction
   - 8.7. Long-Term Reference in H.264 and HEVC
   - 8.8. Emphasis MAP
   - 8.9. NVENC Output in Video Memory
   - 8.10. Alpha Layer Encoding support in HEVC
   - 8.11. Temporal Scalable Video Coding (SVC) in

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

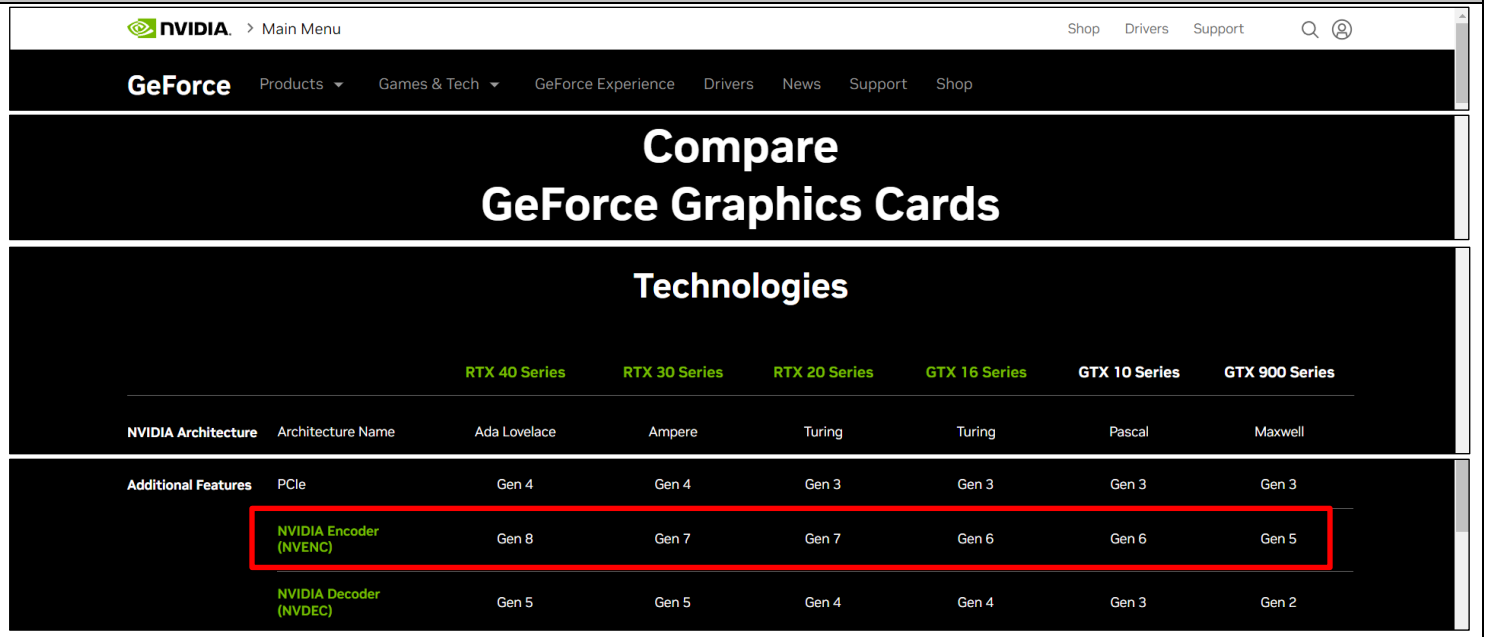
Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP`.
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS`.
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap` (which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL`.

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

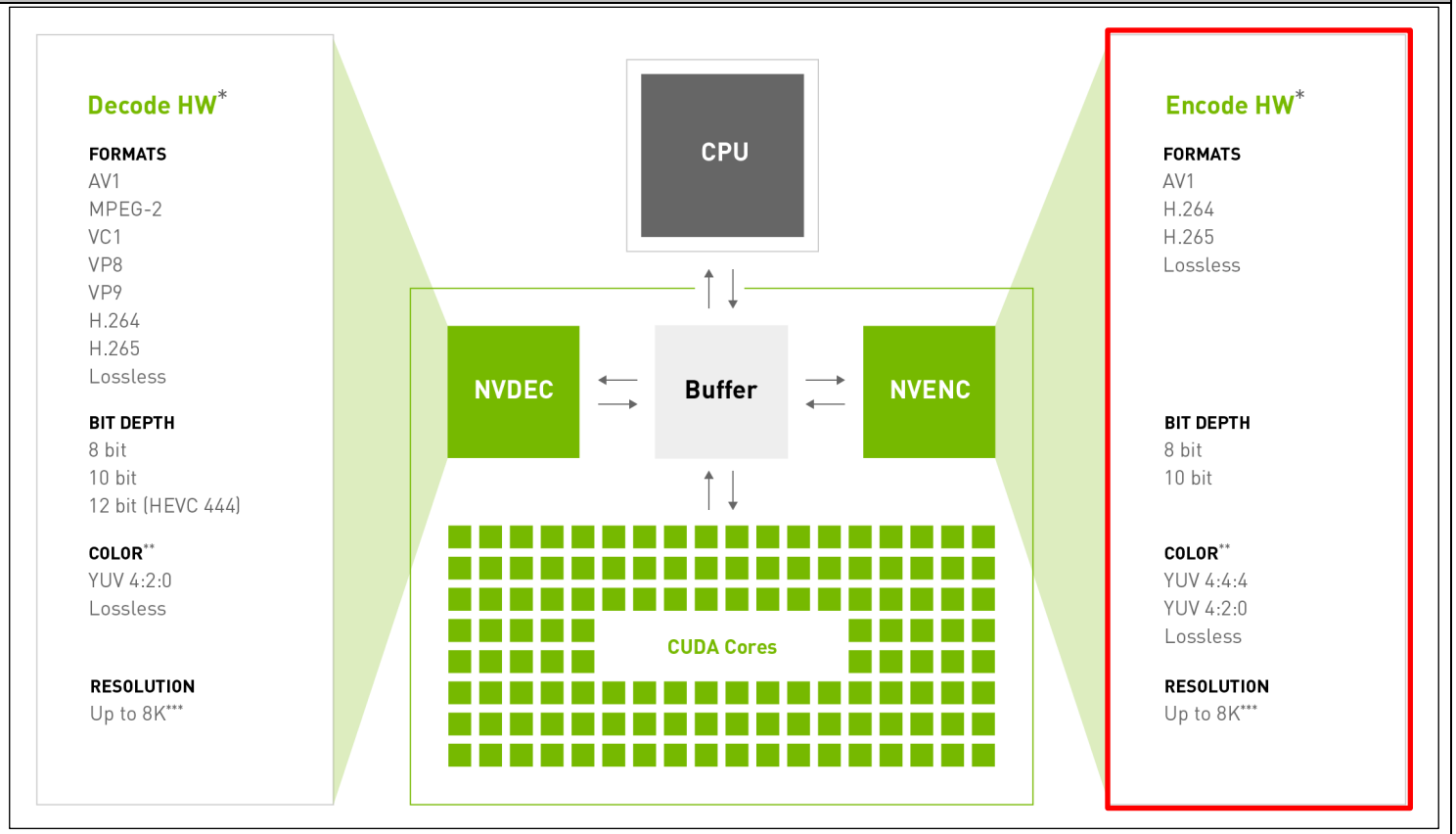
https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

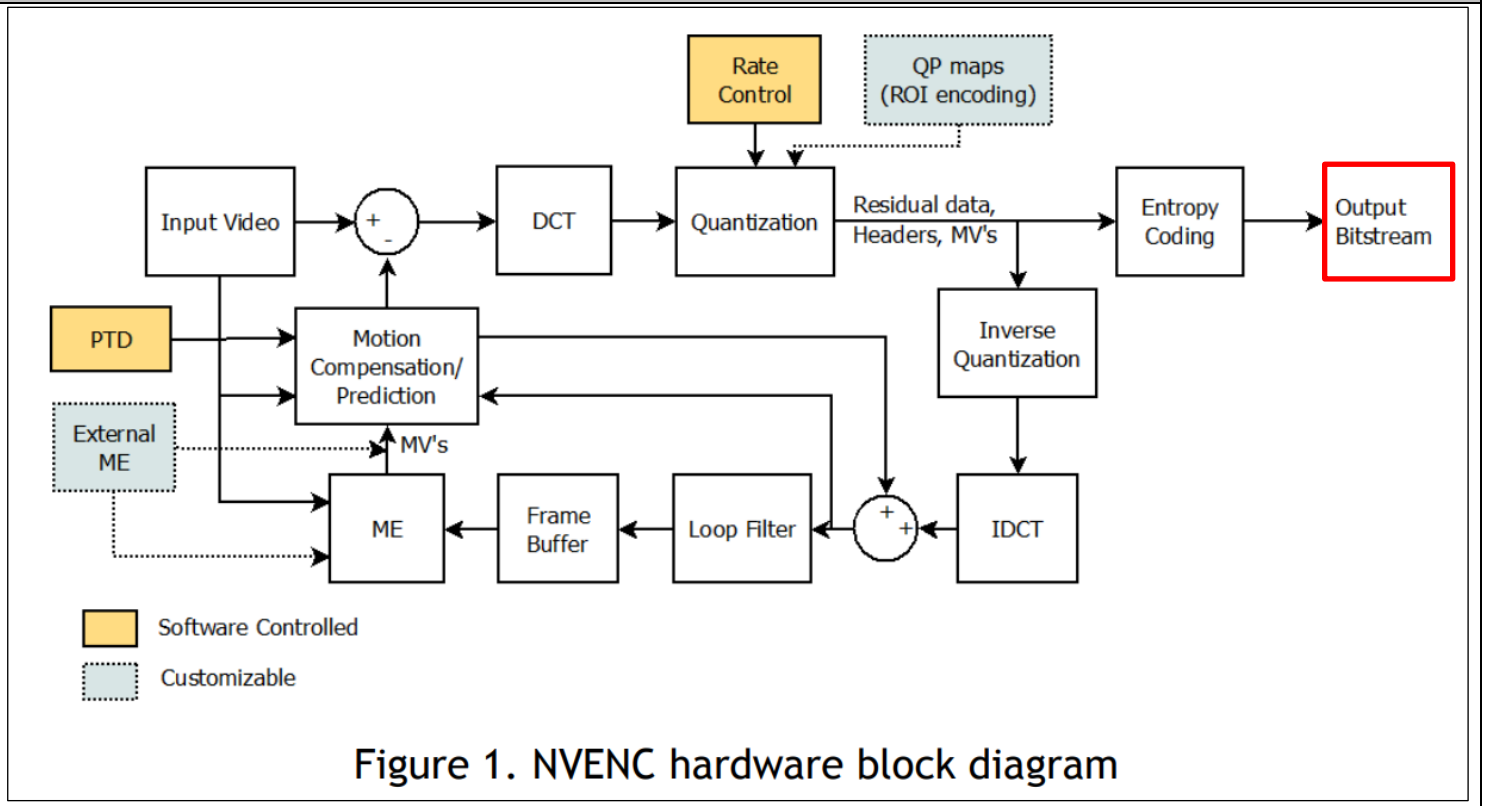


https://www.nvidia.com/en-us/geforce/graphics-cards/compare/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://developer.nvidia.com/sites/default/files/akamai/designworks/images-capture/nv-proviz-encode-decode-diagram-2461893.png

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Figure 1. NVENC hardware block diagram https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1A] an emphasis-level setter for	The Samsung Accused Products comprise: An emphasis-level setter for setting an emphasis level to the input video signal in accordance with at least one factor among control data carried by the input video signal, a

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
setting an emphasis level to the input video signal in accordance with at least one factor among control data carried by the input video signal, a picture state detected from the input video signal and encoding conditions for the input video signal;	picture state detected from the input video signal and encoding conditions for the input video signal. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasis-level setter (*i.e.*, logical unit controlled by NVENC API for defining a QP (quantization parameter) delta map ("Emphasis Map")) for setting an emphasis level (*i.e.*, QP delta values across the map) to the input video signal (*i.e.*, creating custom QP maps for region of interest (ROI) encoding) in accordance with at least one factor among control data carried by the input video signal, a picture state detected from the input video signal and encoding conditions for the input video signal (*e.g.*, QPs derived from the Rate Control algorithm, identification of "high detail"/complex areas).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3. NVENC BLOCK DIAGRAM

Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.

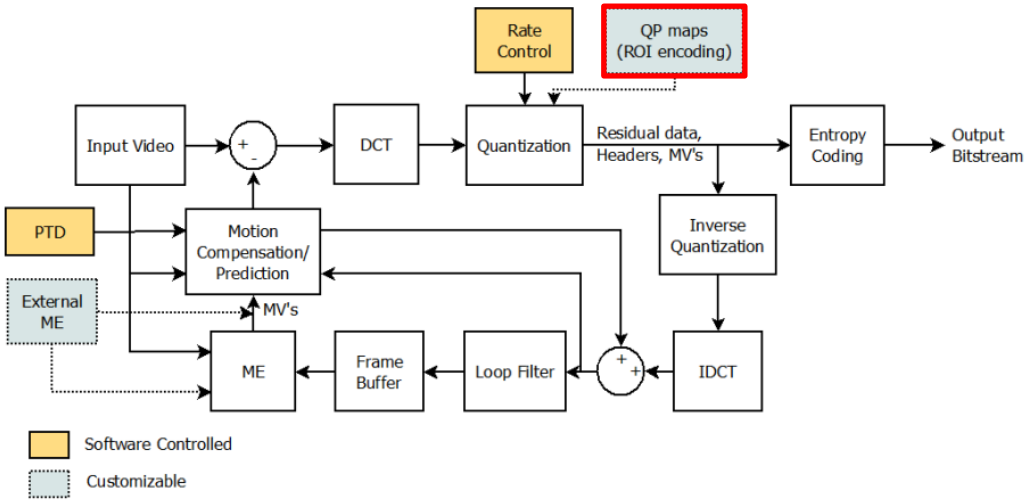
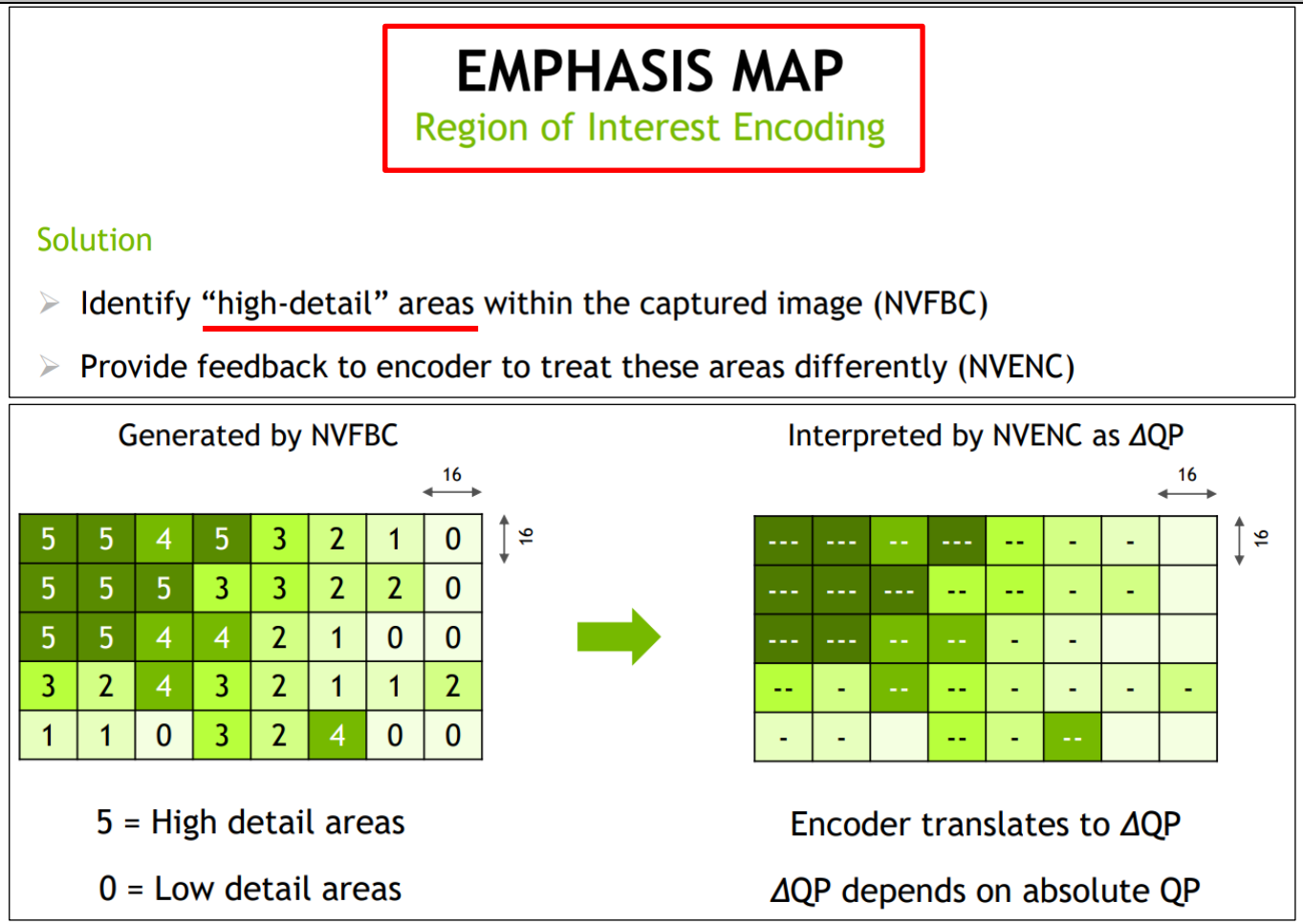


Figure 1. NVENC hardware block diagram

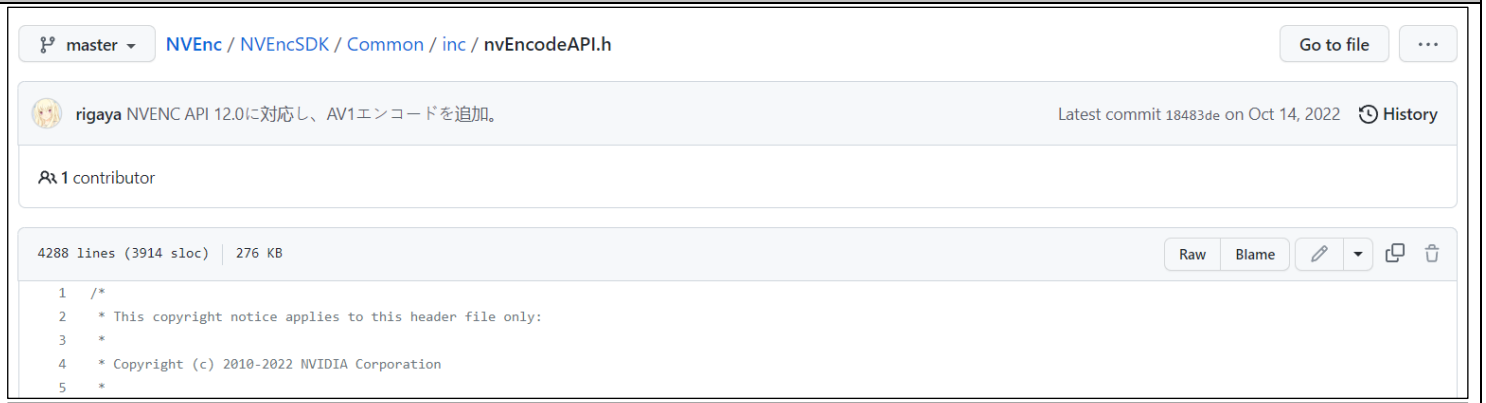
https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
336 /**
337  * QP MAP MODE
338  */
339 typedef enum _NV_ENC_QP_MAP_MODE
340 {
341     NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
342     NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
343     NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
344     NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
345 } NV_ENC_QP_MAP_MODE;
346
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

The NVEncode API Emphasis Map feature specifies regions in the frame to be encoded at different levels of quality (*i.e.*, high-complexity areas at higher quality (at a lower QP)). Subsequently, an adjustment (*i.e.*, emphasis level) is applied to the QP used to encode the respective macroblock. A value of said adjustment depends on an absolute value of the QP as decided by the rate control algorithm and the actual emphasis level for the respective macroblock (*i.e.*, in accordance with at least one factor among control data carried by the input video signal). The picture state is detected from the input video signal and encoding conditions for the input video signal, as the client has "prior knowledge of the image complexity" from the input video signal:

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Multi pass frame encoding

When determining the QP to use for encoding a frame, it is beneficial if NVENC knows the overall complexity of the frame to distribute the available bit budget in the most optimal manner. In some situations, multi-pass encoding may also help catch larger motion between frames. For this purpose, NVENC supports the following types of multi-pass frame encoding modes:

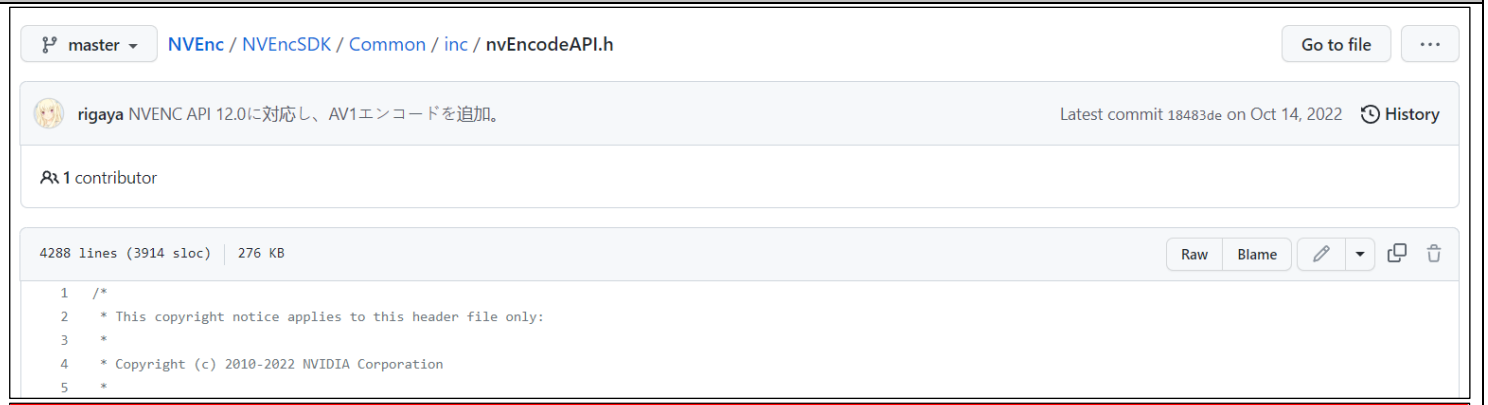
- 1-pass per frame encoding (`NV_ENC_MULTI_PASS_DISABLED`)
- 2-passes per frame, with first pass in quarter resolution and second pass in full resolution (`NV_ENC_TWO_PASS_QUARTER_RESOLUTION`)
- 2-passes per frame, with both passes in full resolution (`NV_ENC_TWO_PASS_FULL_RESOLUION`).

In 1-pass rate control modes, NVENC estimates the required QP for the macroblock and immediately encodes the macroblock. In 2-pass rate control modes, NVENC estimates the complexity of the frame to be encoded and determines bit distribution across the frame in the first pass. In the second pass, NVENC encodes macroblocks in the frame using the distribution determined in the first pass. As a result, with 2-pass rate control modes, NVENC can distribute the bits more optimally within the frame and can reach closer to the target bitrate, especially for CBR encoding. Note, however, that everything else being the same, performance of 2-pass rate control mode is lower than that of 1-pass rate control mode. The client application should choose an appropriate multi-pass rate control mode after evaluating various modes, as each of the modes has its own advantages and disadvantages. `NV_ENC_TWO_PASS_FULL_RESOLUION` generates better statistics for the second pass, whereas `NV_ENC_TWO_PASS_QUARTER_RESOLUTION` results in larger motion vectors being caught and fed as hints to second pass.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

NVEncode API includes a class: "NV_ENC_RC_PARAMS::qpMapMode" which includes Emphasis mode (*i.e.*, NV_ENC_QP_MAP_EMPHASIS) as one of the map modes. In Emphasis mode, "qpDeltaMap" specifies the Emphasis Map Level per macroblock. The map level along with QP for a particular macroblock, selected by rate control, is used to compute a QP modifier. Different Emphasis map levels are associated with different respective Delta QP values. For example, zero Delta QP value is associated with Emphasis Map Level 0. Similarly, very high Delta QP value is associated with Emphasis Map Level 5.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

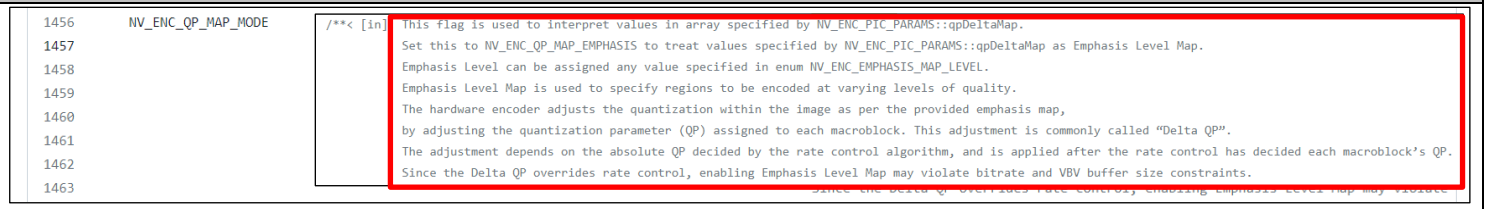
4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
323  /**
324   * Emphasis Levels
325   */
326  typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
327  {
328      NV_ENC_EMPHASIS_MAP_LEVEL_0              = 0x0,      /**< Emphasis Map Level 0, for zero Delta QP value */
329      NV_ENC_EMPHASIS_MAP_LEVEL_1              = 0x1,      /**< Emphasis Map Level 1, for very low Delta QP value */
330      NV_ENC_EMPHASIS_MAP_LEVEL_2              = 0x2,      /**< Emphasis Map Level 2, for low Delta QP value */
331      NV_ENC_EMPHASIS_MAP_LEVEL_3              = 0x3,      /**< Emphasis Map Level 3, for medium Delta QP value */
332      NV_ENC_EMPHASIS_MAP_LEVEL_4              = 0x4,      /**< Emphasis Map Level 4, for high Delta QP value */
333      NV_ENC_EMPHASIS_MAP_LEVEL_5              = 0x5       /**< Emphasis Map Level 5, for very high Delta QP value */
334  } NV_ENC_EMPHASIS_MAP_LEVEL;
335
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

```
1416   typedef struct _NV_ENC_RC_PARAMS
1417   {
1418      uint32_t                     version;
1419      NV_ENC_PARAMS_RC_MODE        rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1B] an emphasizer for applying the emphasis processing to the input video signal at the emphasis level to obtain an emphasized video signal;	The Samsung Accused Products comprise: An emphasizer for applying the emphasis processing to the input video signal at the emphasis level to obtain an emphasized video signal. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasizer (*i.e.*, NVENC encoder) for applying the emphasis processing (*i.e.*, supports the hardware function of region-of-interest encoding) to the input video signal at the emphasis level (*i.e.*, adjusted QP values, Delta QP) to obtain an emphasized (*e.g.*, quantized) video signal.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3. NVENC BLOCK DIAGRAM

Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.

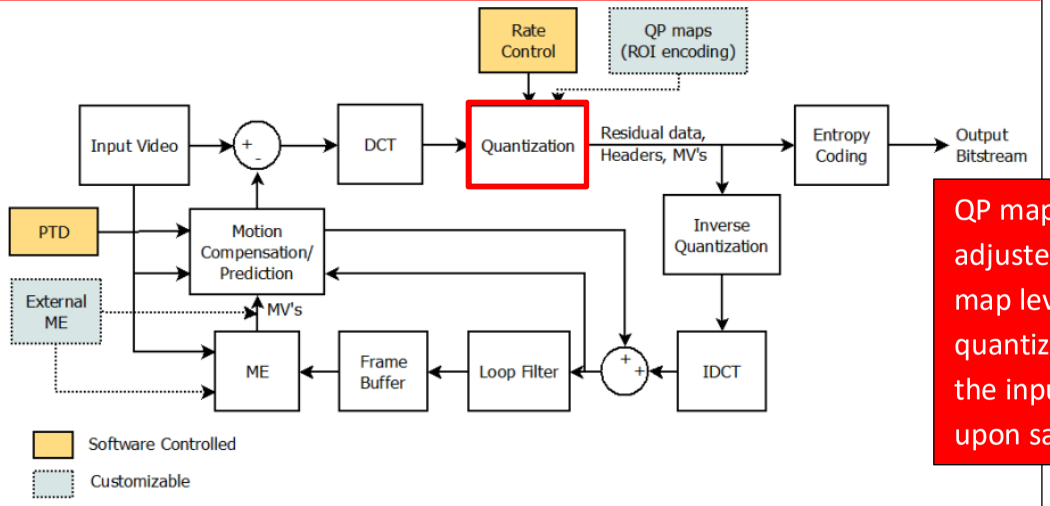


Figure 1. NVENC hardware block diagram

QP maps, including QP values adjusted with respective emphasis map levels, are provided to the quantization block which quantizes the input from the DCT block based upon said adjusted QP values.

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

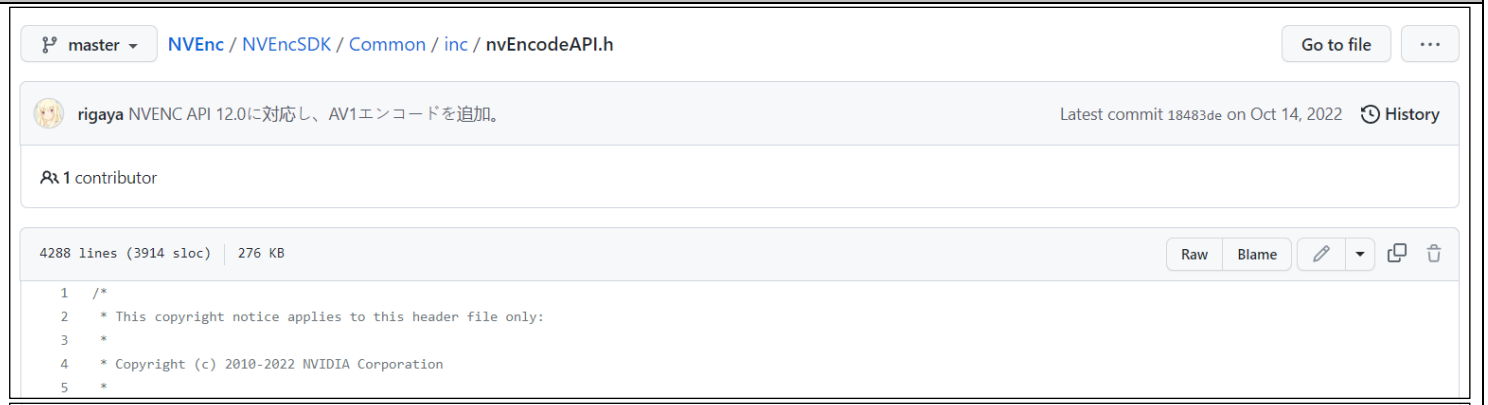
Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

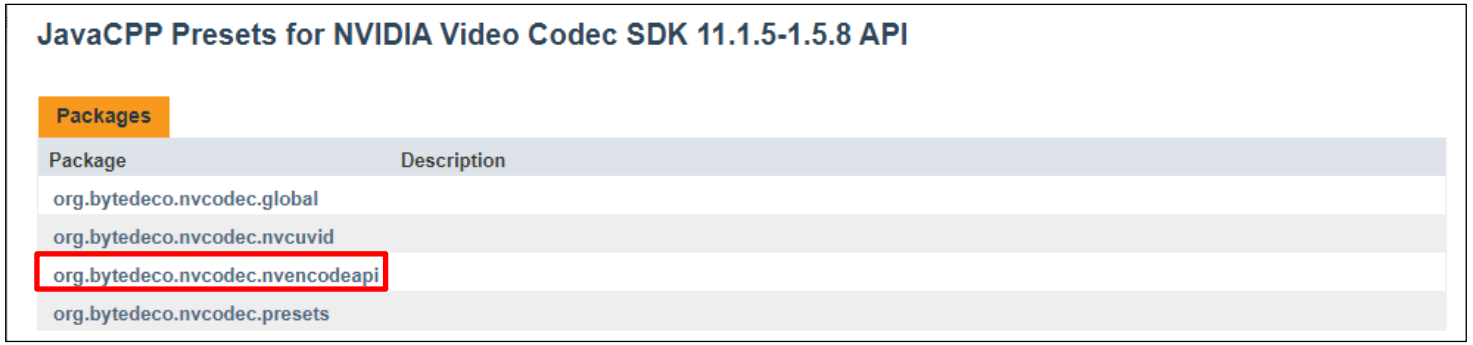
1 contributor

4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
336  /**
337   * QP MAP MODE
338   */
339  typedef enum _NV_ENC_QP_MAP_MODE
340  {
341      NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
342      NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
343      NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
344      NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
345  } NV_ENC_QP_MAP_MODE;
346
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h



JavaCPP Presets for NVIDIA Video Codec SDK 11.1.5-1.5.8 API

Packages

Package	Description
org.bytedeco.nvcodec.global	
org.bytedeco.nvcodec.nvcuvid	
org.bytedeco.nvcodec.nvencodeapi	
org.bytedeco.nvcodec.presets	

http://bytedeco.org/javacpp-presets/nvcodec/apidocs/overview-summary.html

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

org.bytedeco.nvcodec.nvencodeapi

Class NV_ENC_PIC_PARAMS

...

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description

...

qpDeltaMap

```
@Cast(value="int8_t*")
public BytePointer qpDeltaMap()
```

[in]: Specifies the pointer to signed byte array containing value per MB for H264 and per CTB for HEVC in raster scan order for the current picture, which will be interpreted depending on NV_ENC_RC_PARAMS::qpMapMode. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_DELTA, qpDeltaMap specifies QP modifier per MB for H264 and per CTB for HEVC. This QP modifier will be applied on top of the QP chosen by rate control. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_EMPHASIS, qpDeltaMap specifies Emphasis Level Map per MB for H264. This level value along with QP chosen by rate control is used to compute the QP modifier, which in turn is applied on top of QP chosen by rate control. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_DISABLED, value in qpDeltaMap will be ignored.

http://bytedeco.org/javacpp-presets/nvcodec/apidocs/org/bytedeco/nvcodec/nvencodeapi/NV_ENC_PIC_PARAMS.html

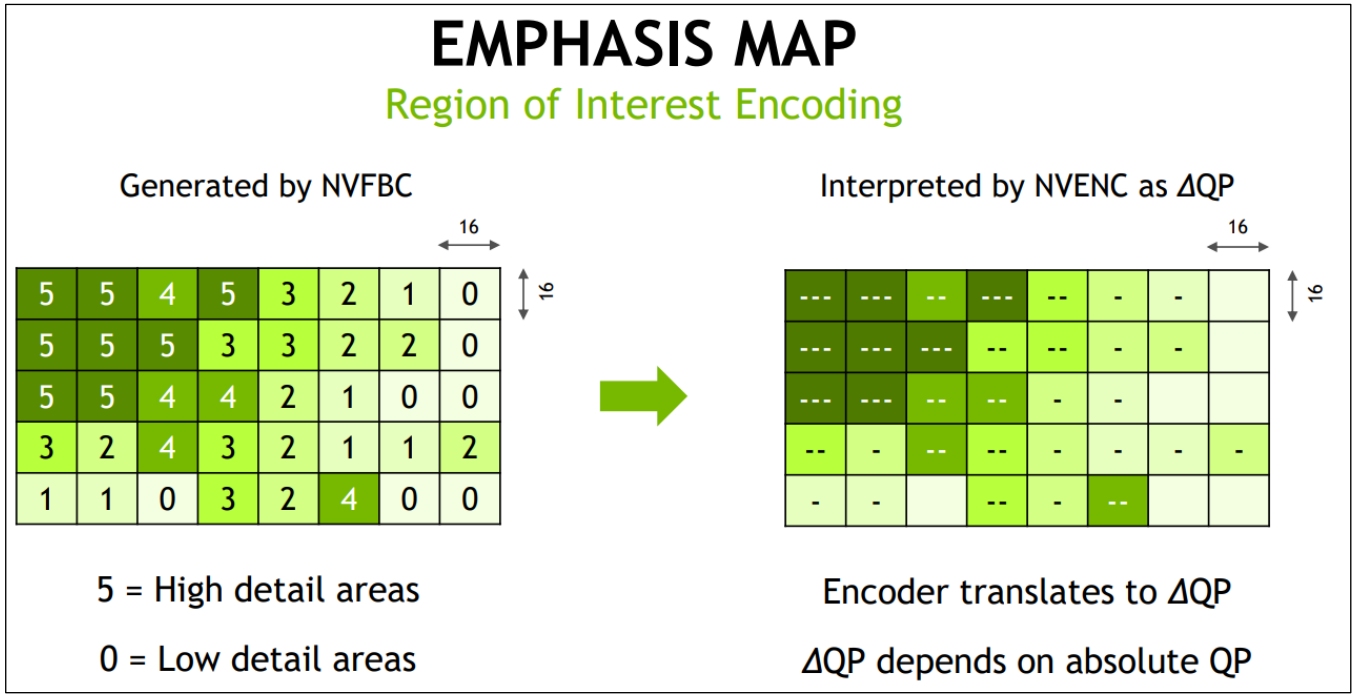
```
1416    typedef struct _NV_ENC_RC_PARAMS
1417    {
1418        uint32_t                    version;
1419        NV_ENC_PARAMS_RC_MODE       rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

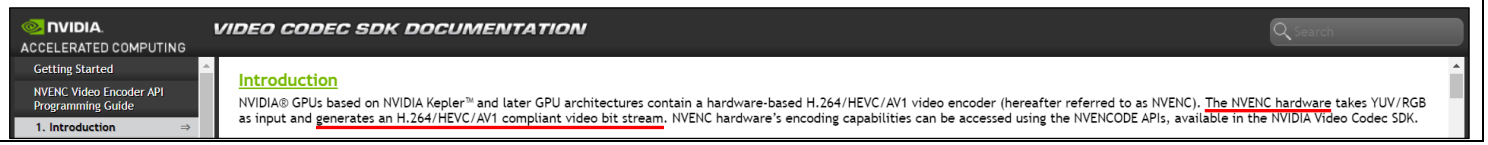
```
1456        NV_ENC_QP_MAP_MODE      /**< [in]
1457
1458
1459
1460
1461
1462
1463
```

This flag is used to interpret values in array specified by NV_ENC_PIC_PARAMS::qpDeltaMap.
Set this to NV_ENC_QP_MAP_EMPHASIS to treat values specified by NV_ENC_PIC_PARAMS::qpDeltaMap as Emphasis Level Map.
Emphasis Level can be assigned any value specified in enum NV_ENC_EMPHASIS_MAP_LEVEL.
Emphasis Level Map is used to specify regions to be encoded at varying levels of quality.
The hardware encoder adjusts the quantization within the image as per the provided emphasis map,
by adjusting the quantization parameter (QP) assigned to each macroblock. This adjustment is commonly called "Delta QP".
The adjustment depends on the absolute QP decided by the rate control algorithm, and is applied after the rate control has decided each macroblock's QP.
Since the Delta QP overrides rate control, enabling Emphasis Level Map may violate bitrate and VBV buffer size constraints.

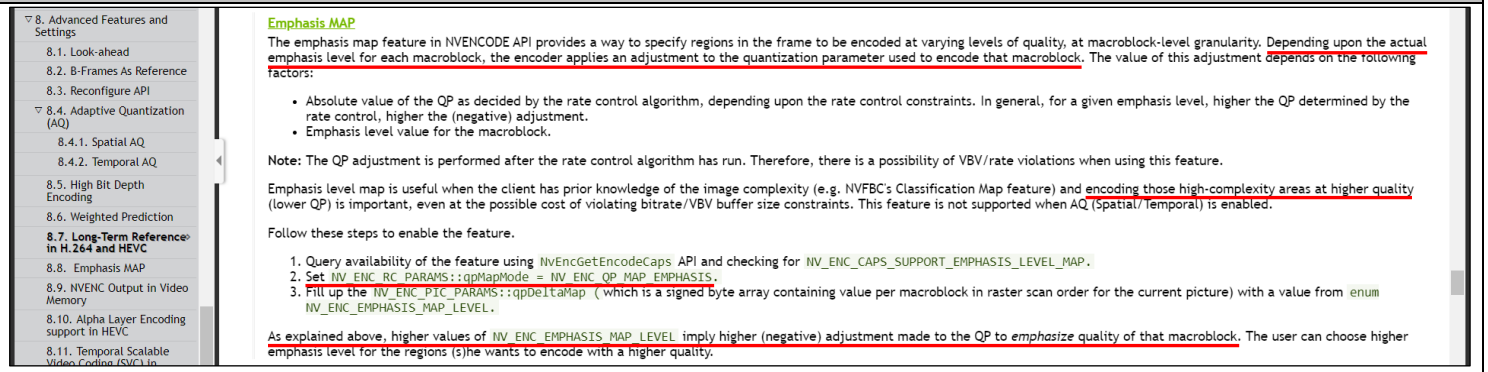
https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h



https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1C] an encoder for encoding the emphasized video signal to obtain a video bitstream; and	The Samsung Accused Products comprise: an encoder for encoding the emphasized video signal to obtain a video bitstream. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises a video-emphasis encoding apparatus for applying an emphasis processing to an input video signal to obtain a video bitstream. NVENC encoder, based on the information of the regions specified by NVEncode API, applies an adjustment to the quantization parameter used to encode a macroblock for obtaining a video bitstream, depending on an emphasis level for the respective macroblock (*i.e.*, an encoder for encoding the emphasized video signal to obtain a video bitstream). For example, higher values of emphasis map level would imply higher adjustment made to the QP to emphasize the quality of the respective macroblock. 

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

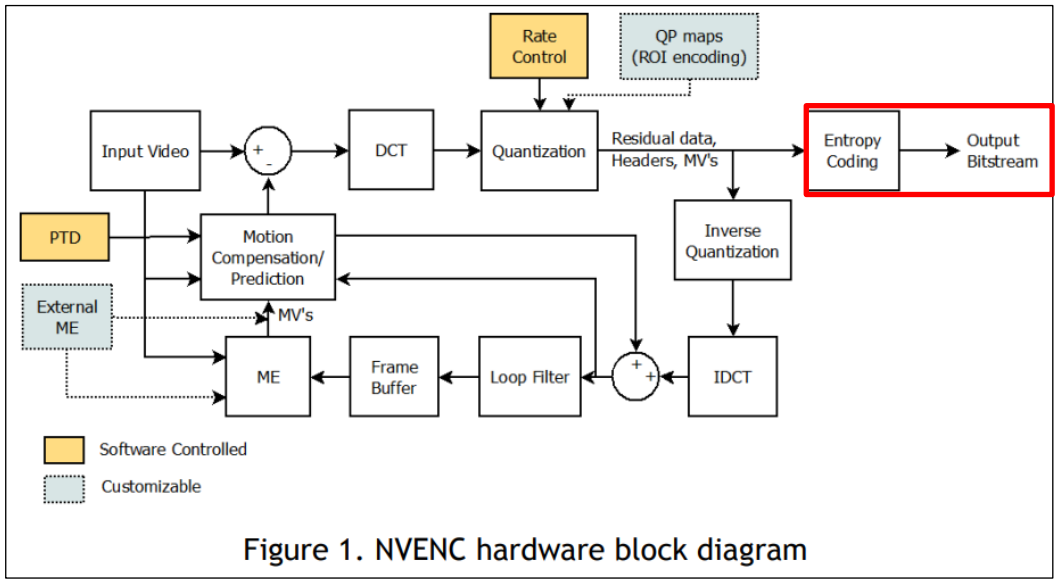


Figure 1. NVENC hardware block diagram

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1D] a multiplexer for multiplexing the video bitstream and data on the emphasis level.	The Samsung Accused Products comprise: A multiplexer for multiplexing the video bitstream and data on the emphasis level. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises a multiplexer for multiplexing the video bitstream and data on the emphasis level (*e.g.*, *mb_qp_delta*).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

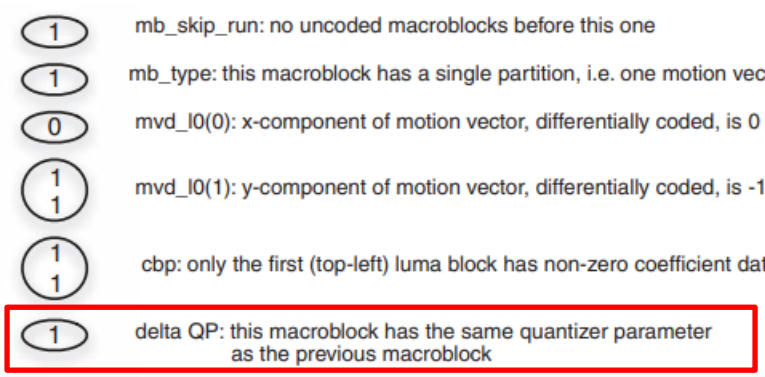
Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ⋯

```
323  /**
324   * Emphasis Levels
325   */
326  typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
327  {
328      NV_ENC_EMPHASIS_MAP_LEVEL_0              = 0x0,       /**< Emphasis Map Level 0, for zero Delta QP value */
329      NV_ENC_EMPHASIS_MAP_LEVEL_1              = 0x1,       /**< Emphasis Map Level 1, for very low Delta QP value */
330      NV_ENC_EMPHASIS_MAP_LEVEL_2              = 0x2,       /**< Emphasis Map Level 2, for low Delta QP value */
331      NV_ENC_EMPHASIS_MAP_LEVEL_3              = 0x3,       /**< Emphasis Map Level 3, for medium Delta QP value */
332      NV_ENC_EMPHASIS_MAP_LEVEL_4              = 0x4,       /**< Emphasis Map Level 4, for high Delta QP value */
333      NV_ENC_EMPHASIS_MAP_LEVEL_5              = 0x5        /**< Emphasis Map Level 5, for very high Delta QP value */
334  } NV_ENC_EMPHASIS_MAP_LEVEL;
335
336  /**
337   * QP MAP MODE
338   */
339  typedef enum _NV_ENC_QP_MAP_MODE
340  {
341      NV_ENC_QP_MAP_DISABLED               = 0x0,             /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
342      NV_ENC_QP_MAP_EMPHASIS               = 0x1,             /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
343      NV_ENC_QP_MAP_DELTA                  = 0x2,             /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
344      NV_ENC_QP_MAP                        = 0x3,             /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
345  } NV_ENC_QP_MAP_MODE;
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

4.6 The H.264 Syntax

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Example An H.264 decoder receives the following series of bits, describing an inter-coded macroblock or P MB coded using context-adaptive VLCs: 11011111000111000111 0010 (alternate syntax elements highlighted). Figure 4.20 shows the syntax elements in this sequence of bits. The header elements, the first eight bits and six syntax elements, indicate the macroblock type (P, one motion vector),  https://last.hit.bme.hu/download/vidtech/k%C3%B6nyvek/Iain%20E.%20Richardson%20-%20H264%20(2nd%20edition).pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3.5 Entropy coder

The entropy encoder converts a series of symbols representing elements of the video sequence into a compressed bitstream suitable for transmission or storage. Input symbols may include quantized transform coefficients, run-level or zerotree encoded as described in section 3.4.4, motion vectors with integer or sub-pixel resolution, marker codes that indicate a resynchronization point in the sequence, macroblock headers, picture headers, sequence headers etc and supplementary information, 'side' information that is not essential for correct decoding.

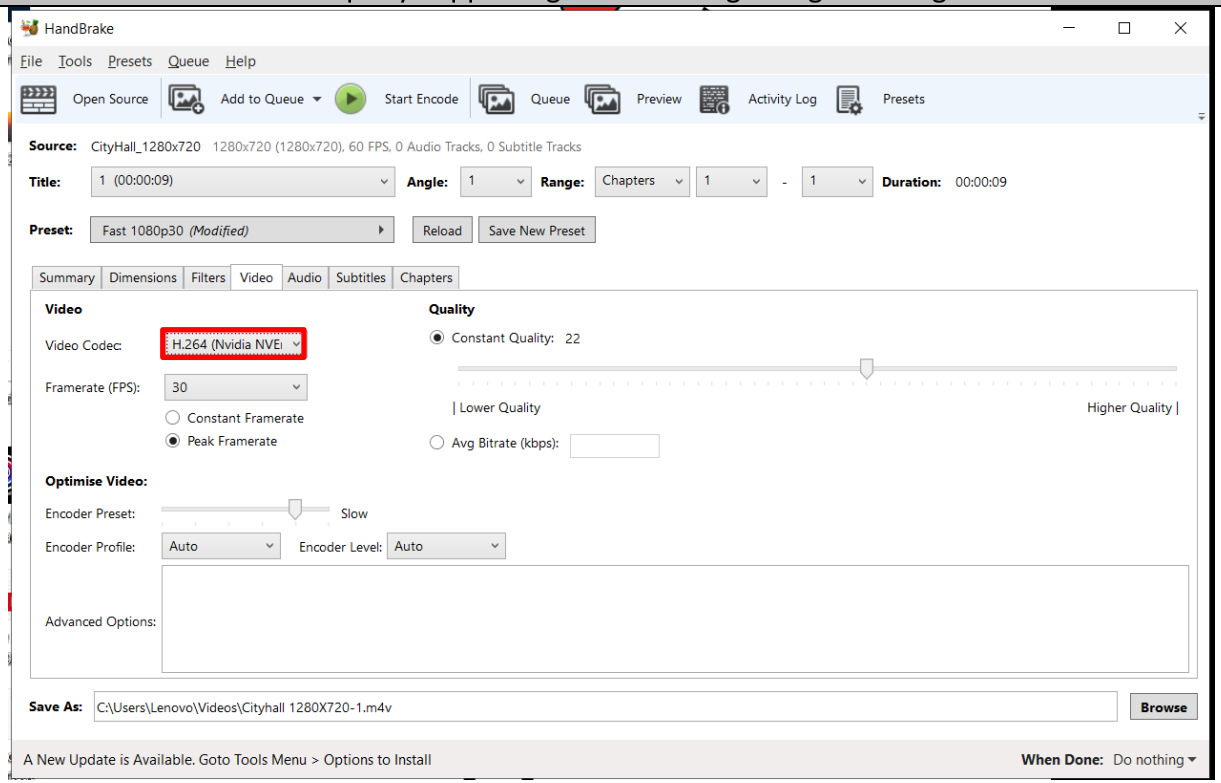
3.5.1 Predictive coding

...

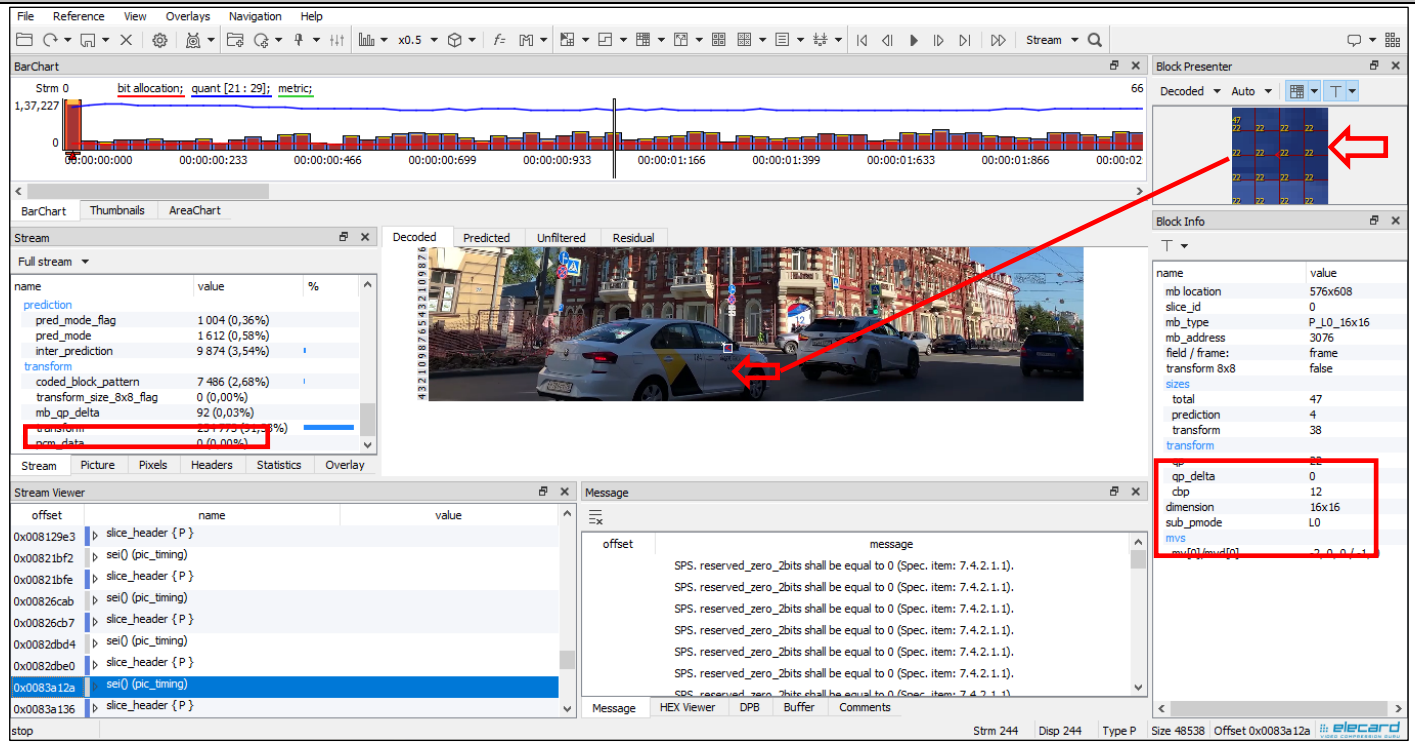
The quantization parameter or quantizer step size controls the trade-off between compression efficiency and image quality. In a real-time video CODEC it may be necessary to modify the quantization within an encoded frame, for example to change the compression ratio in order to match the coded bit rate to a transmission channel rate. It is usually sufficient to change the parameter by a small amount between successive coded macroblocks. The modified quantization parameter must be signalled to the decoder and instead of sending a new quantization parameter value, it may be preferable to send a delta or difference value, e.g. +/−1 or +/−2, indicating the change required. Fewer bits are required to encode a small delta value than to encode a completely new quantization parameter.

https://last.hit.bme.hu/download/vidtech/k%C3%B6nyvek/Iain%20E.%20Richardson%20-%20H264%20(2nd%20edition).pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



Source: Internal Testing

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Internal Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
2. The video-emphasis encoding	The Samsung Accused Products comprise: a detector for detecting a degree of quantization applied to the input video signal, wherein the emphasis-level setter sets the emphasis level in accordance with the detected degree

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
apparatus according to claim 1 further comprising a detector for detecting a degree of quantization applied to the input video signal, wherein the emphasis-level setter sets the emphasis level in accordance with the detected degree of quantization as the encoding conditions.	of quantization as the encoding conditions. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasis-level setter (*i.e.*, logical unit controlled by NVENC API for detecting a degree of quantization (*e.g.*, quantization parameters that indicate a region of interest or a region of high detail). The emphasis-level setter sets an emphasis level (*i.e.*, QP delta values across the map) to the input video signal (*i.e.*, creating custom QP maps for region of interest (ROI) encoding) based on the degree of quantization.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3. NVENC BLOCK DIAGRAM

Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.

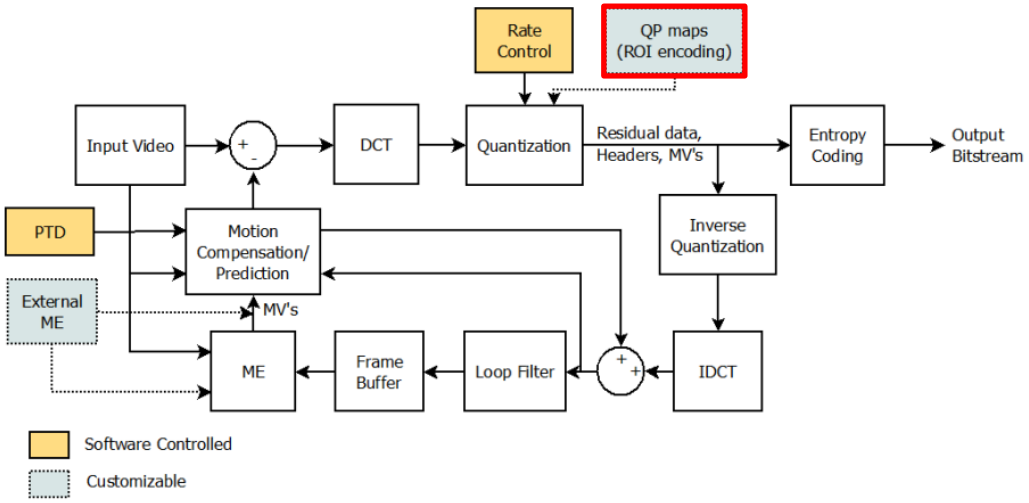


Figure 1. NVENC hardware block diagram

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

EMPHASIS MAP

Region of Interest Encoding

Solution

- Identify "high-detail" areas within the captured image (NVFBC)
- Provide feedback to encoder to treat these areas differently (NVENC)

Generated by NVFBC

5	5	4	5	3	2	1	0
5	5	5	3	3	2	2	0
5	5	4	4	2	1	0	0
3	2	4	3	2	1	1	2
1	1	0	3	2	4	0	0

16 × 16

5 = High detail areas

0 = Low detail areas

Interpreted by NVENC as ΔQP

---	---	--	---	--	-	-	
---	---	---	--	--	-	-	
---	---	--	--	-	-		
--	-	--	--	-	-	-	-
-	-		--	-	--		

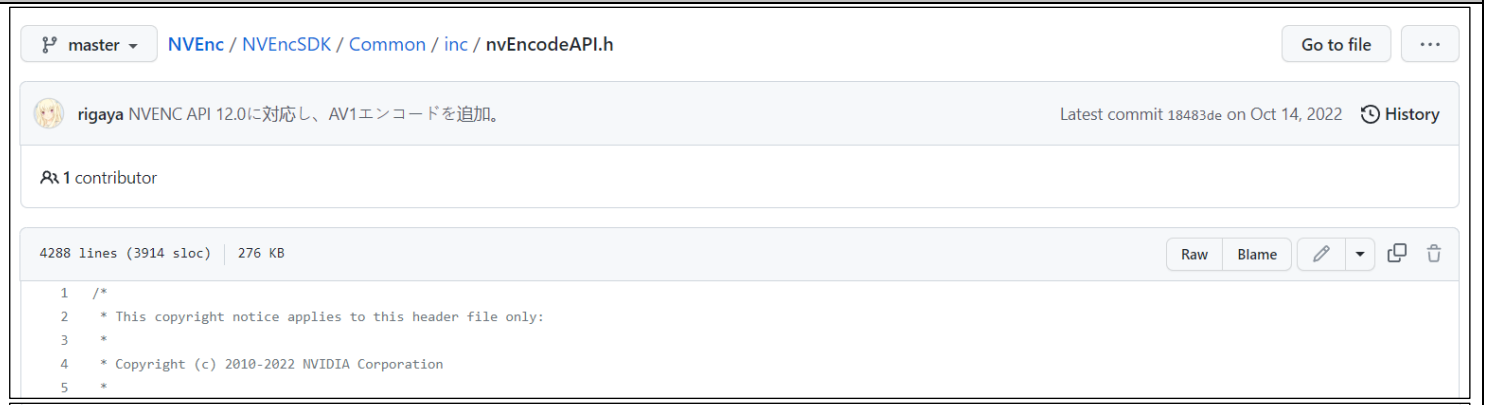
16 × 16

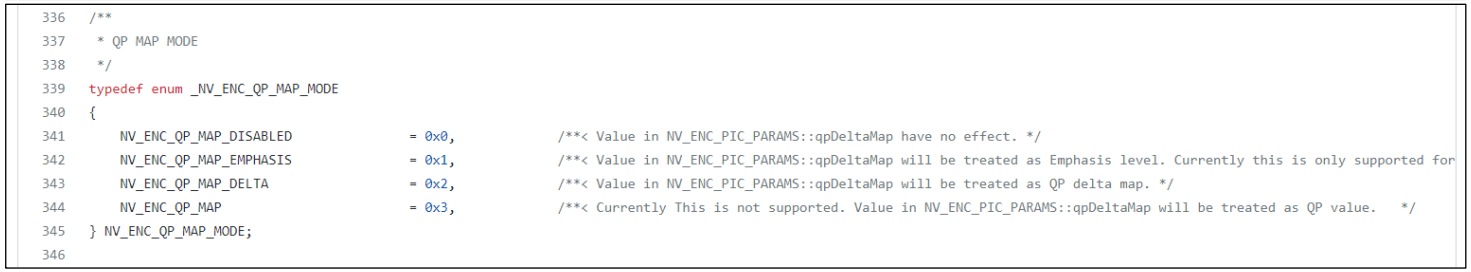
Encoder translates to ΔQP

ΔQP depends on absolute QP

https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products





```
/**
 * QP MAP MODE
 */
typedef enum _NV_ENC_QP_MAP_MODE
{
    NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
    NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
    NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
    NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
} NV_ENC_QP_MAP_MODE;
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

The NVEncode API Emphasis Map feature specifies regions in the frame to be encoded at different levels of quality (*i.e.*, high-complexity areas at higher quality (at a lower QP)). Subsequently, an adjustment (*i.e.*, emphasis level) is applied to the QP used to encode the respective macroblock. A value of said adjustment depends on an absolute value of the QP as decided by the rate control algorithm and the actual emphasis level for the respective macroblock (*i.e.*, in accordance with at least one factor among control data carried by the input video signal). The picture state is detected from the input video signal and encoding conditions for the input video signal, as the client has "prior knowledge of the image complexity" from the input video signal:

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Multi pass frame encoding

When determining the QP to use for encoding a frame, it is beneficial if NVENC knows the overall complexity of the frame to distribute the available bit budget in the most optimal manner. In some situations, multi-pass encoding may also help catch larger motion between frames. For this purpose, NVENC supports the following types of multi-pass frame encoding modes:

- 1-pass per frame encoding (`NV_ENC_MULTI_PASS_DISABLED`)
- 2-passes per frame, with first pass in quarter resolution and second pass in full resolution (`NV_ENC_TWO_PASS_QUARTER_RESOLUTION`)
- 2-passes per frame, with both passes in full resolution (`NV_ENC_TWO_PASS_FULL_RESOLUION`).

In 1-pass rate control modes, NVENC estimates the required QP for the macroblock and immediately encodes the macroblock. In 2-pass rate control modes, NVENC estimates the complexity of the frame to be encoded and determines bit distribution across the frame in the first pass. In the second pass, NVENC encodes macroblocks in the frame using the distribution determined in the first pass. As a result, with 2-pass rate control modes, NVENC can distribute the bits more optimally within the frame and can reach closer to the target bitrate, especially for CBR encoding. Note, however, that everything else being the same, performance of 2-pass rate control mode is lower than that of 1-pass rate control mode. The client application should choose an appropriate multi-pass rate control mode after evaluating various modes, as each of the modes has its own advantages and disadvantages. `NV_ENC_TWO_PASS_FULL_RESOLUION` generates better statistics for the second pass, whereas `NV_ENC_TWO_PASS_QUARTER_RESOLUTION` results in larger motion vectors being caught and fed as hints to second pass.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

NVEncode API includes a class: "NV_ENC_RC_PARAMS::qpMapMode" which includes Emphasis mode (*i.e.*, NV_ENC_QP_MAP_EMPHASIS) as one of the map modes. In Emphasis mode, "qpDeltaMap" specifies the Emphasis Map Level per macroblock. The map level along with QP for a particular macroblock, selected by rate control, is used to compute a QP modifier. Different Emphasis map levels are associated with different respective Delta QP values. For example, zero Delta QP value is associated with Emphasis Map Level 0. Similarly, very high Delta QP value is associated with Emphasis Map Level 5.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

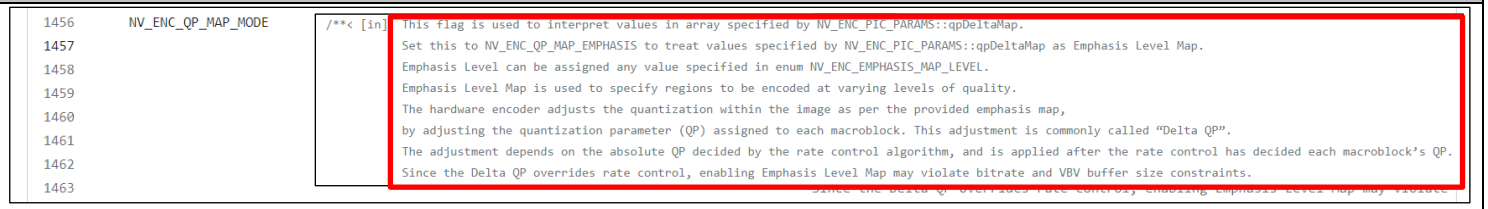
4288 lines (3914 sloc) | 276 KB Raw Blame

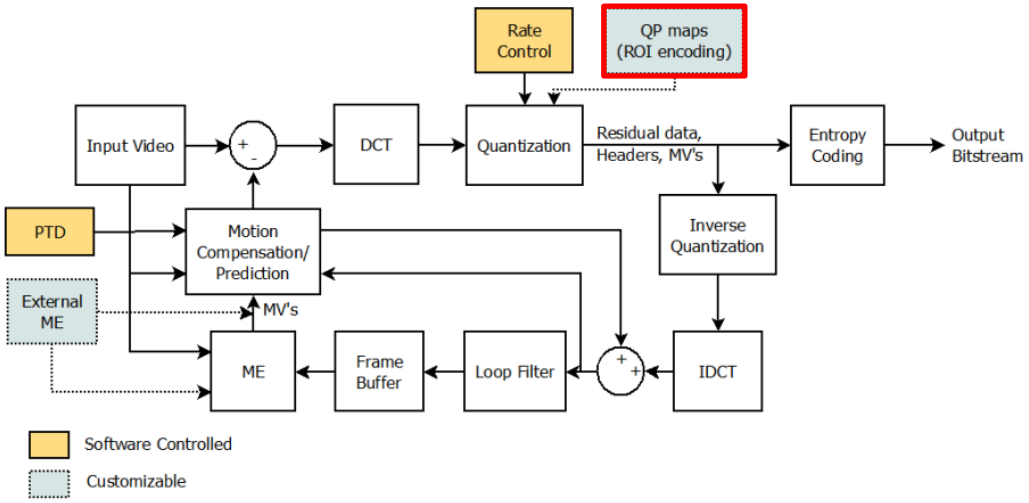
```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
323  /**
324   * Emphasis Levels
325   */
326  typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
327  {
328      NV_ENC_EMPHASIS_MAP_LEVEL_0              = 0x0,       /**< Emphasis Map Level 0, for zero Delta QP value */
329      NV_ENC_EMPHASIS_MAP_LEVEL_1              = 0x1,       /**< Emphasis Map Level 1, for very low Delta QP value */
330      NV_ENC_EMPHASIS_MAP_LEVEL_2              = 0x2,       /**< Emphasis Map Level 2, for low Delta QP value */
331      NV_ENC_EMPHASIS_MAP_LEVEL_3              = 0x3,       /**< Emphasis Map Level 3, for medium Delta QP value */
332      NV_ENC_EMPHASIS_MAP_LEVEL_4              = 0x4,       /**< Emphasis Map Level 4, for high Delta QP value */
333      NV_ENC_EMPHASIS_MAP_LEVEL_5              = 0x5        /**< Emphasis Map Level 5, for very high Delta QP value */
334  } NV_ENC_EMPHASIS_MAP_LEVEL;
335
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

```
1416   typedef struct _NV_ENC_RC_PARAMS
1417   {
1418      uint32_t                     version;
1419      NV_ENC_PARAMS_RC_MODE        rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
3. The video-emphasis encoding apparatus according to claim 1 further comprising a detector for detecting a degree of enhancement effects already applied to the input video signal, wherein the emphasis-level setter sets	The Samsung Accused Products comprise: a detector for detecting a degree of quantization applied to the input video signal, wherein the emphasis-level setter sets the emphasis level in accordance with the detected degree of quantization as the encoding conditions. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the Emphasis Map feature. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasis-level setter (*i.e.*, logical unit controlled by NVENC API for defining a QP (quantization parameter) delta map ("Emphasis Map")) for setting an emphasis level (*i.e.*, QP delta values across the map) to the input video signal (*i.e.*, creating custom QP maps for region of interest (ROI) encoding) based on a detector within the emphasis-level setter detecting a degree of enhancement effects applied to the input video signal (*e.g.*, QPs derived from the Rate Control algorithm or QPs identifying "high detail"/complex areas that indicate a degree of enhancement in the input video signal).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the emphasis level in accordance with the detected degree of enhancement effects as the picture state detected from the input video signal.	3. NVENC BLOCK DIAGRAM Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.  Figure 1. NVENC hardware block diagram https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

EMPHASIS MAP

Region of Interest Encoding

Solution

- Identify "high-detail" areas within the captured image (NVFBC)
- Provide feedback to encoder to treat these areas differently (NVENC)

Generated by NVFBC

5	5	4	5	3	2	1	0
5	5	5	3	3	2	2	0
5	5	4	4	2	1	0	0
3	2	4	3	2	1	1	2
1	1	0	3	2	4	0	0

16 × 16

5 = High detail areas

0 = Low detail areas

Interpreted by NVENC as ΔQP

---	---	--	---	--	-	-	
---	---	---	--	--	-	-	
---	---	--	--	-	-		
--	-	--	--	-	-	-	-
-	-		--	-	--		

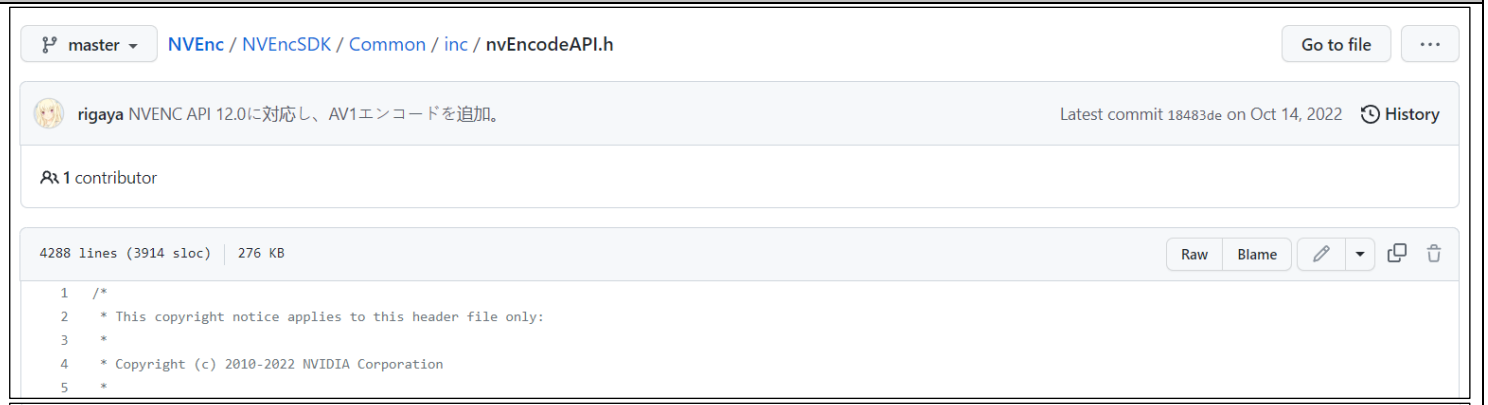
16 × 16

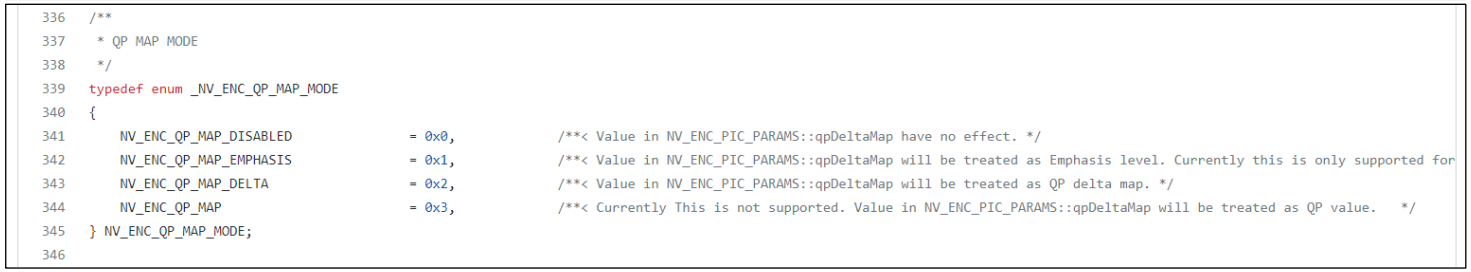
Encoder translates to ΔQP

ΔQP depends on absolute QP

https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products





```
/**
 * QP MAP MODE
 */
typedef enum _NV_ENC_QP_MAP_MODE
{
    NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
    NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
    NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
    NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
} NV_ENC_QP_MAP_MODE;
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

The NVEncode API Emphasis Map feature specifies regions in the frame to be encoded at different levels of quality (*i.e.*, high-complexity areas at higher quality (at a lower QP)). Subsequently, an adjustment (*i.e.*, emphasis level) is applied to the QP used to encode the respective macroblock. A value of said adjustment depends on an absolute value of the QP as decided by the rate control algorithm and the actual emphasis level for the respective macroblock (*i.e.*, in accordance with at least one factor among control data carried by the input video signal). The picture state is detected from the input video signal and encoding conditions for the input video signal, as the client has "prior knowledge of the image complexity" from the input video signal:

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Multi pass frame encoding

When determining the QP to use for encoding a frame, it is beneficial if NVENC knows the overall complexity of the frame to distribute the available bit budget in the most optimal manner. In some situations, multi-pass encoding may also help catch larger motion between frames. For this purpose, NVENC supports the following types of multi-pass frame encoding modes:

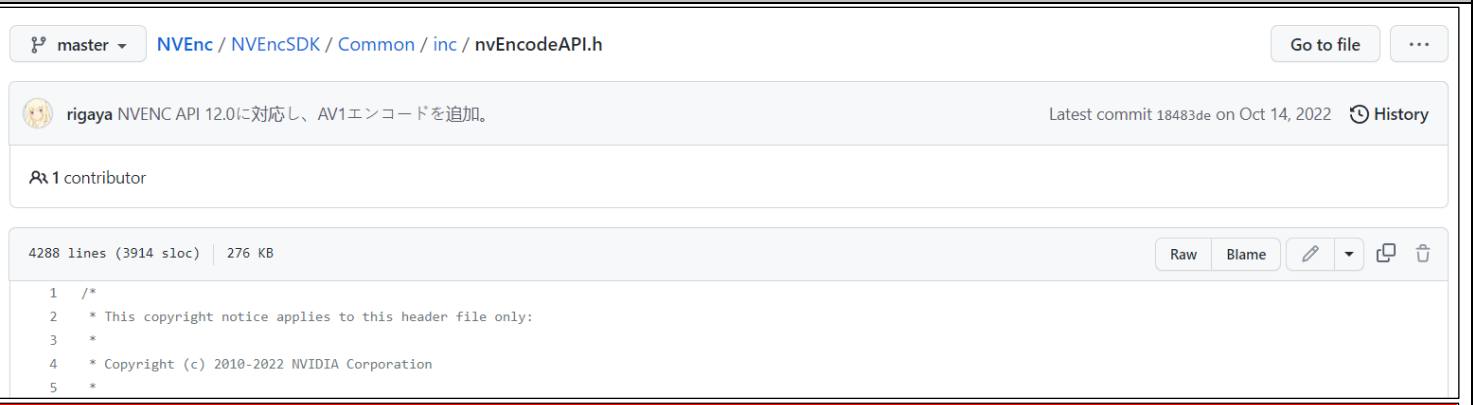
- 1-pass per frame encoding (`NV_ENC_MULTI_PASS_DISABLED`)
- 2-passes per frame, with first pass in quarter resolution and second pass in full resolution (`NV_ENC_TWO_PASS_QUARTER_RESOLUTION`)
- 2-passes per frame, with both passes in full resolution (`NV_ENC_TWO_PASS_FULL_RESOLUION`).

In 1-pass rate control modes, NVENC estimates the required QP for the macroblock and immediately encodes the macroblock. In 2-pass rate control modes, NVENC estimates the complexity of the frame to be encoded and determines bit distribution across the frame in the first pass. In the second pass, NVENC encodes macroblocks in the frame using the distribution determined in the first pass. As a result, with 2-pass rate control modes, NVENC can distribute the bits more optimally within the frame and can reach closer to the target bitrate, especially for CBR encoding. Note, however, that everything else being the same, performance of 2-pass rate control mode is lower than that of 1-pass rate control mode. The client application should choose an appropriate multi-pass rate control mode after evaluating various modes, as each of the modes has its own advantages and disadvantages. `NV_ENC_TWO_PASS_FULL_RESOLUION` generates better statistics for the second pass, whereas `NV_ENC_TWO_PASS_QUARTER_RESOLUTION` results in larger motion vectors being caught and fed as hints to second pass.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

NVEncode API includes a class: "NV_ENC_RC_PARAMS::qpMapMode" which includes Emphasis mode (*i.e.*, NV_ENC_QP_MAP_EMPHASIS) as one of the map modes. In Emphasis mode, "qpDeltaMap" specifies the Emphasis Map Level per macroblock. The map level along with QP for a particular macroblock, selected by rate control, is used to compute a QP modifier. Different Emphasis map levels are associated with different respective Delta QP values. For example, zero Delta QP value is associated with Emphasis Map Level 0. Similarly, very high Delta QP value is associated with Emphasis Map Level 5.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

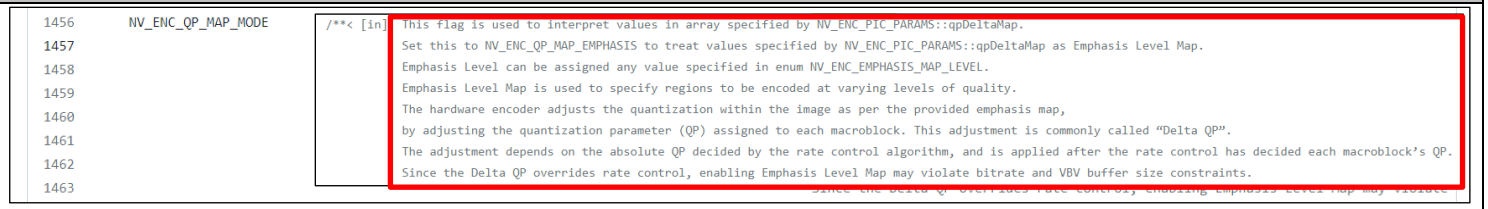
4288 lines (3914 sloc) | 276 KB Raw Blame

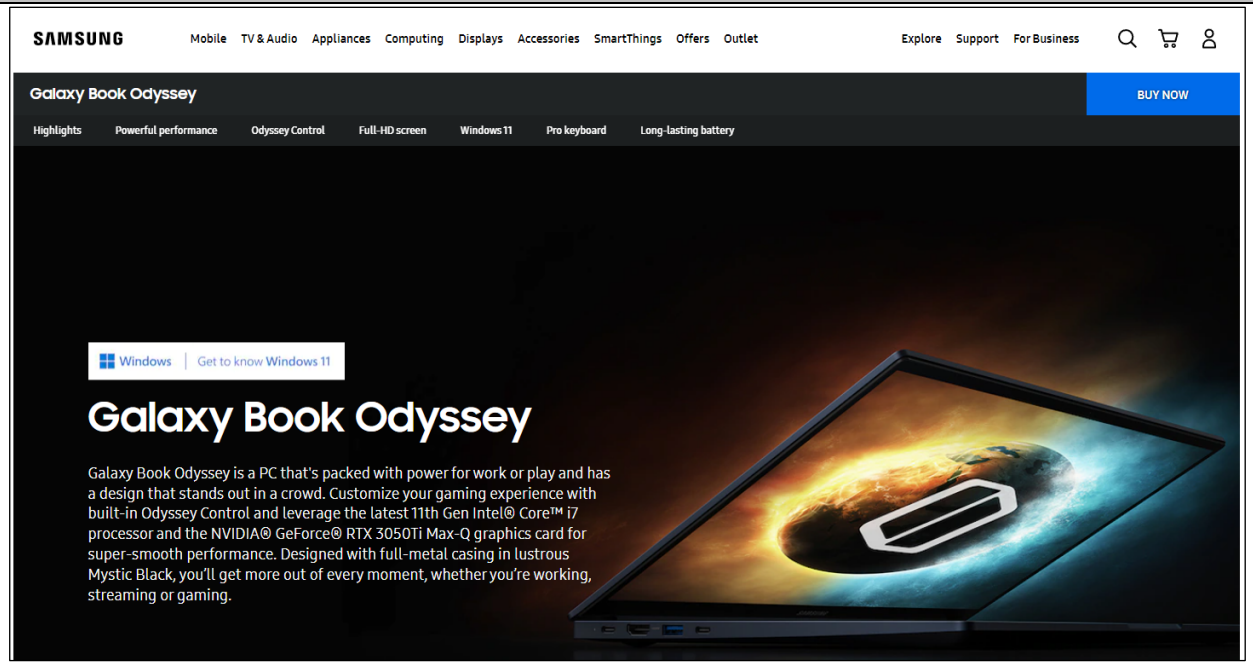
```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

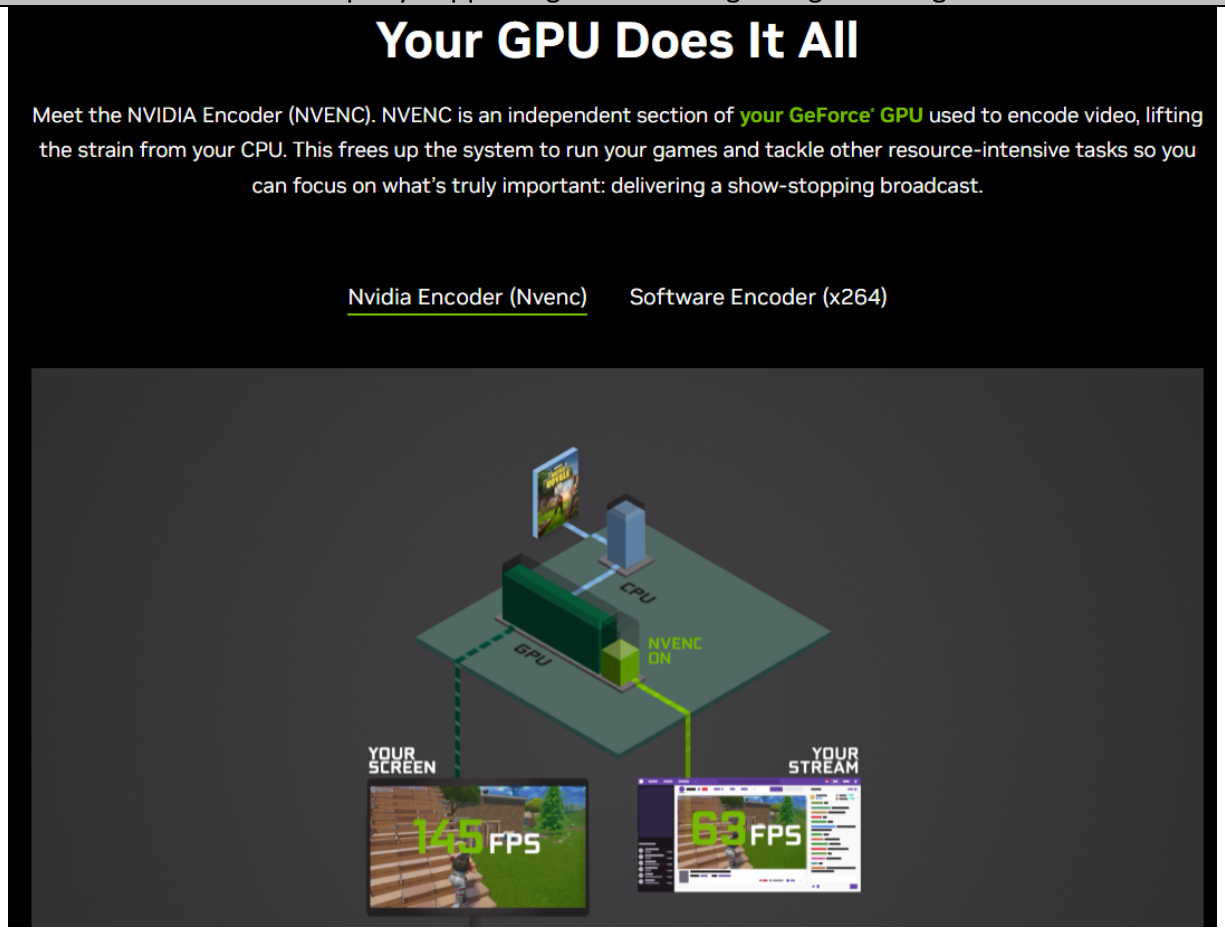
```
323  /**
324   * Emphasis Levels
325   */
326  typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
327  {
328      NV_ENC_EMPHASIS_MAP_LEVEL_0              = 0x0,       /**< Emphasis Map Level 0, for zero Delta QP value */
329      NV_ENC_EMPHASIS_MAP_LEVEL_1              = 0x1,       /**< Emphasis Map Level 1, for very low Delta QP value */
330      NV_ENC_EMPHASIS_MAP_LEVEL_2              = 0x2,       /**< Emphasis Map Level 2, for low Delta QP value */
331      NV_ENC_EMPHASIS_MAP_LEVEL_3              = 0x3,       /**< Emphasis Map Level 3, for medium Delta QP value */
332      NV_ENC_EMPHASIS_MAP_LEVEL_4              = 0x4,       /**< Emphasis Map Level 4, for high Delta QP value */
333      NV_ENC_EMPHASIS_MAP_LEVEL_5              = 0x5        /**< Emphasis Map Level 5, for very high Delta QP value */
334  } NV_ENC_EMPHASIS_MAP_LEVEL;
335
```

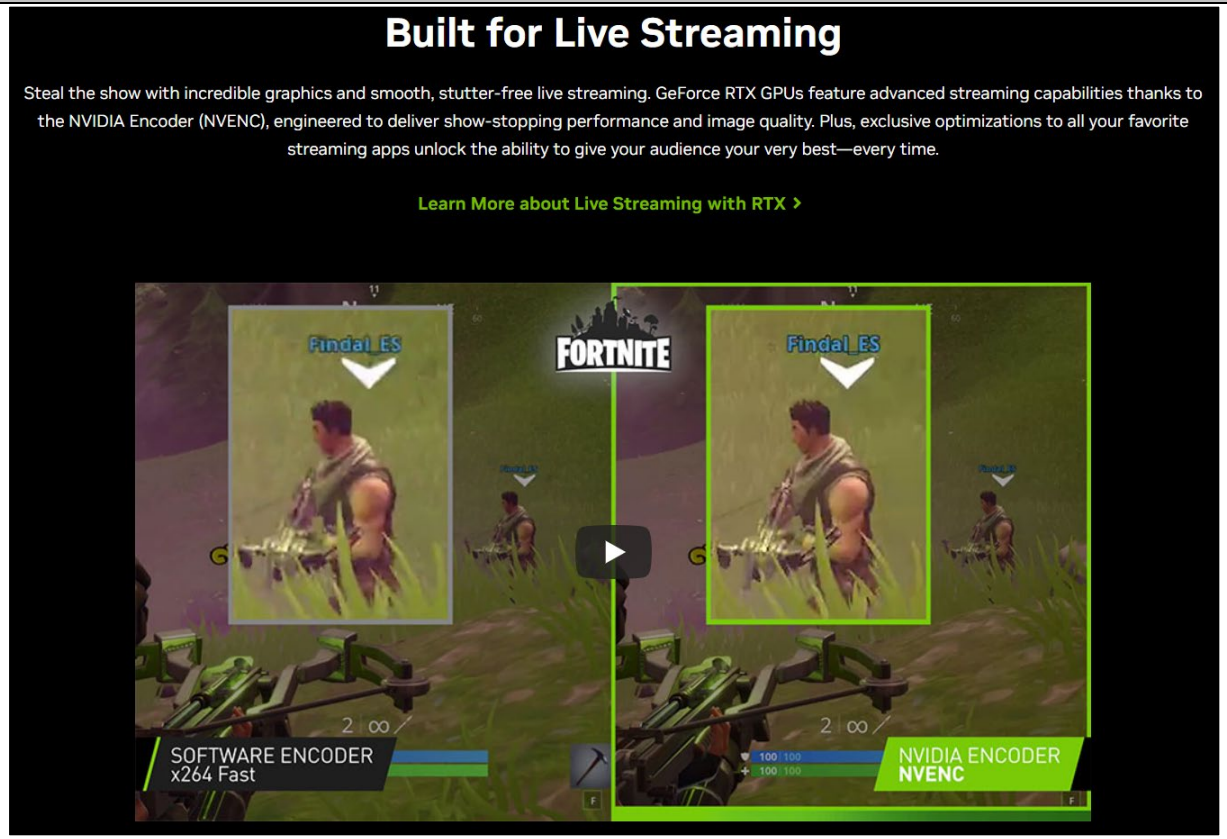
https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

```
1416   typedef struct _NV_ENC_RC_PARAMS
1417   {
1418      uint32_t                     version;
1419      NV_ENC_PARAMS_RC_MODE        rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

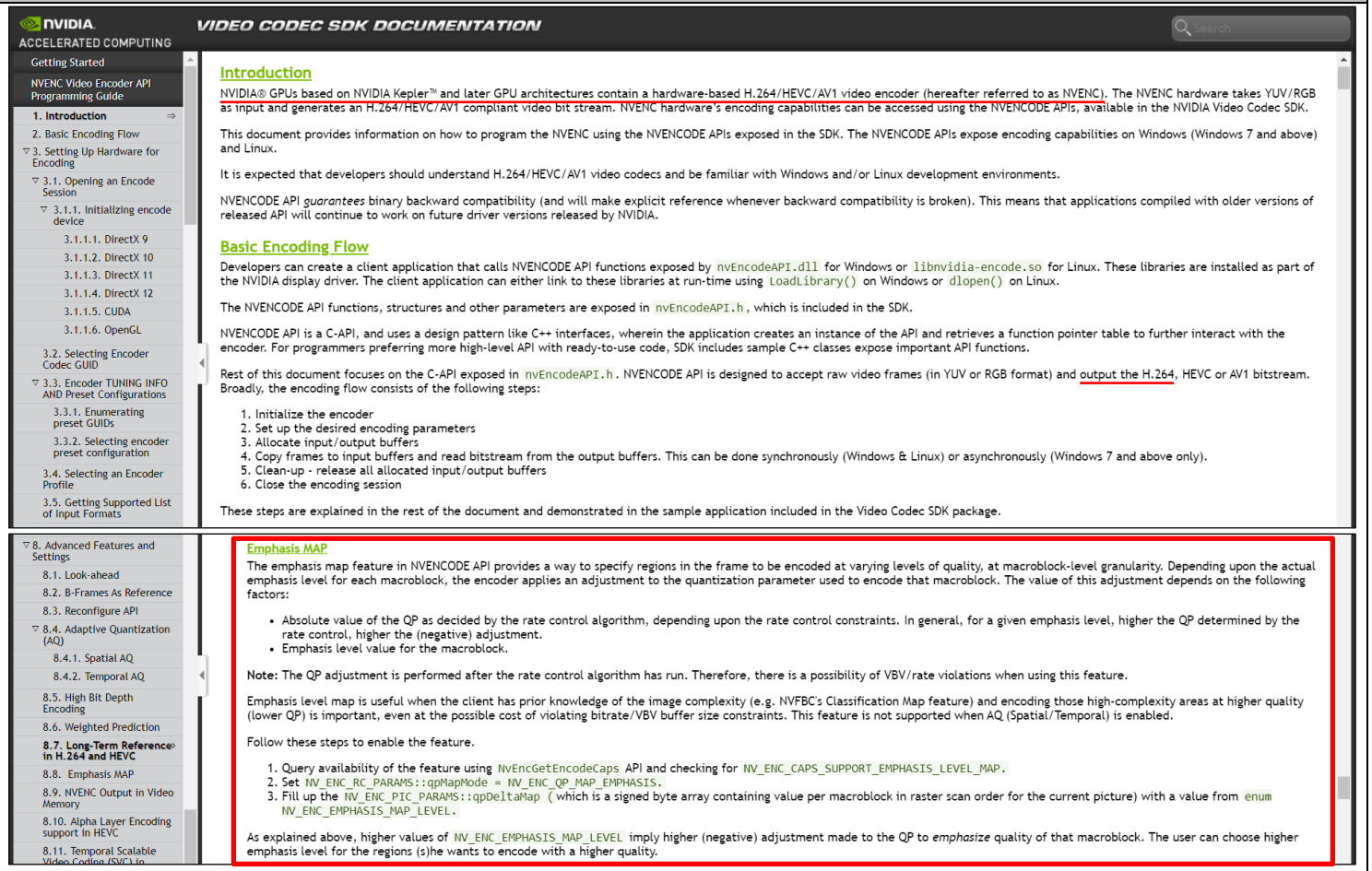
Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4P] A method of video-emphasis encoding for applying an emphasis processing to an input video signal to obtain a video bitstream comprising the steps of:	Samsung or users of the Samsung Accused Products perform: A method of video-emphasis encoding for applying an emphasis processing to an input video signal to obtain a video bitstream. Samsung directly infringes and induces infringement by causing a user to perform the Emphasis Map feature of the Accused Products. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q is a video-emphasis encoding apparatus for applying an emphasis processing to an input video signal (*i.e.*, for applying an emphasis map to an input video signal using the NVIDIA NVENC encoder) to obtain a video bitstream (*e.g.*, an H.264 video bitstream).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.samsung.com/us/computing/galaxy-book-odyssey/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.nvidia.com/en-us/geforce/broadcasting/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.nvidia.com/en-my/geforce/laptops/30-series/#features

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



NVIDIA ACCELERATED COMPUTING — VIDEO CODEC SDK DOCUMENTATION

Getting Started

NVENC Video Encoder API Programming Guide

1. Introduction
2. Basic Encoding Flow
3. Setting Up Hardware for Encoding
   - 3.1. Opening an Encode Session
     - 3.1.1. Initializing encode device
       - 3.1.1.1. DirectX 9
       - 3.1.1.2. DirectX 10
       - 3.1.1.3. DirectX 11
       - 3.1.1.4. DirectX 12
       - 3.1.1.5. CUDA
       - 3.1.1.6. OpenGL
   - 3.2. Selecting Encoder Codec GUID
   - 3.3. Encoder TUNING INFO AND Preset Configurations
     - 3.3.1. Enumerating preset GUIDs
     - 3.3.2. Selecting encoder preset configuration
   - 3.4. Selecting an Encoder Profile
   - 3.5. Getting Supported List of Input Formats

Introduction

NVIDIA® GPUs based on NVIDIA Kepler™ and later GPU architectures contain a hardware-based H.264/HEVC/AV1 video encoder (hereafter referred to as NVENC). The NVENC hardware takes YUV/RGB as input and generates an H.264/HEVC/AV1 compliant video bit stream. NVENC hardware's encoding capabilities can be accessed using the NVENCODE APIs, available in the NVIDIA Video Codec SDK.

This document provides information on how to program the NVENC using the NVENCODE APIs exposed in the SDK. The NVENCODE APIs expose encoding capabilities on Windows (Windows 7 and above) and Linux.

It is expected that developers should understand H.264/HEVC/AV1 video codecs and be familiar with Windows and/or Linux development environments.

NVENCODE API *guarantees* binary backward compatibility (and will make explicit reference whenever backward compatibility is broken). This means that applications compiled with older versions of released API will continue to work on future driver versions released by NVIDIA.

Basic Encoding Flow

Developers can create a client application that calls NVENCODE API functions exposed by `nvEncodeAPI.dll` for Windows or `libnvidia-encode.so` for Linux. These libraries are installed as part of the NVIDIA display driver. The client application can either link to these libraries at run-time using `LoadLibrary()` on Windows or `dlopen()` on Linux.

The NVENCODE API functions, structures and other parameters are exposed in `nvEncodeAPI.h`, which is included in the SDK.

NVENCODE API is a C-API, and uses a design pattern like C++ interfaces, wherein the application creates an instance of the API and retrieves a function pointer table to further interact with the encoder. For programmers preferring more high-level API with ready-to-use code, SDK includes sample C++ classes expose important API functions.

Rest of this document focuses on the C-API exposed in `nvEncodeAPI.h`. NVENCODE API is designed to accept raw video frames (in YUV or RGB format) and output the H.264, HEVC or AV1 bitstream. Broadly, the encoding flow consists of the following steps:

1. Initialize the encoder
2. Set up the desired encoding parameters
3. Allocate input/output buffers
4. Copy frames to input buffers and read bitstream from the output buffers. This can be done synchronously (Windows & Linux) or asynchronously (Windows 7 and above only).
5. Clean-up - release all allocated input/output buffers
6. Close the encoding session

These steps are explained in the rest of the document and demonstrated in the sample application included in the Video Codec SDK package.

8. Advanced Features and Settings
   - 8.1. Look-ahead
   - 8.2. B-Frames As Reference
   - 8.3. Reconfigure API
   - 8.4. Adaptive Quantization (AQ)
     - 8.4.1. Spatial AQ
     - 8.4.2. Temporal AQ
   - 8.5. High Bit Depth Encoding
   - 8.6. Weighted Prediction
   - 8.7. Long-Term Reference in H.264 and HEVC
   - 8.8. Emphasis MAP
   - 8.9. NVENC Output in Video Memory
   - 8.10. Alpha Layer Encoding support in HEVC
   - 8.11. Temporal Scalable Video Coding (SVC) in

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

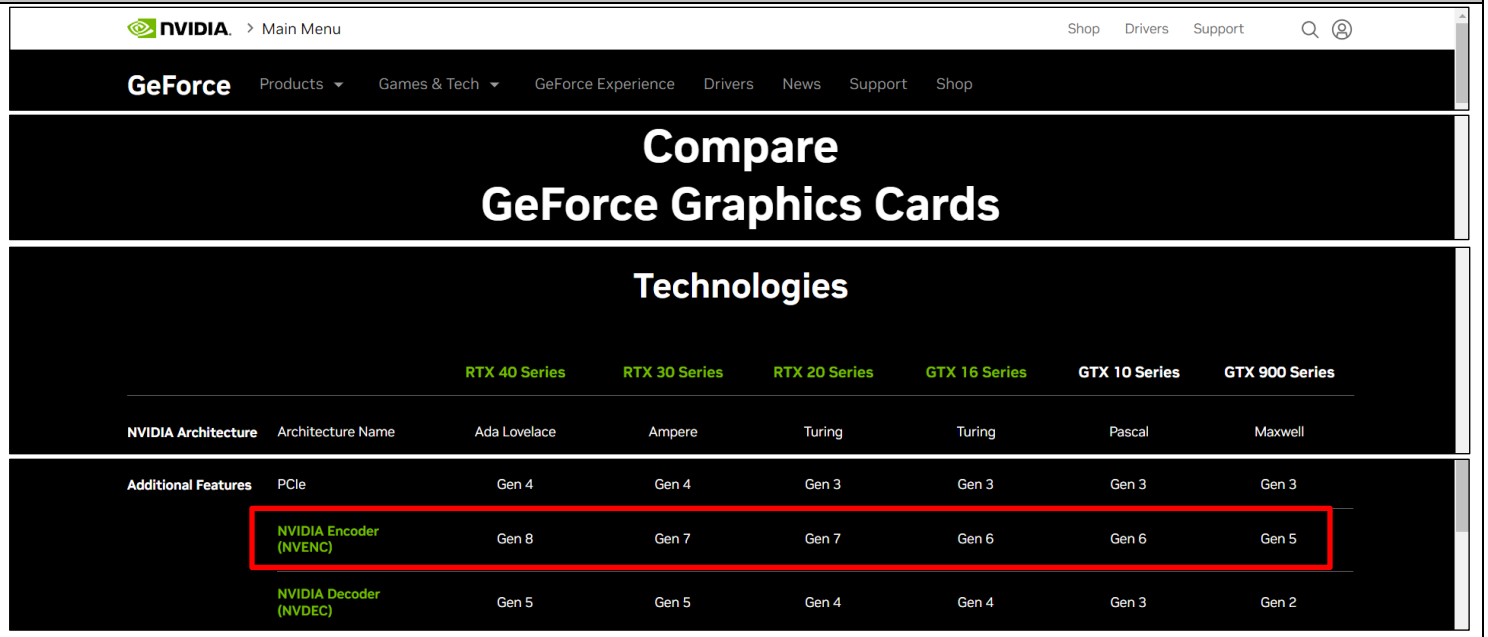
Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP`.
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS`.
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap` (which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL`.

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

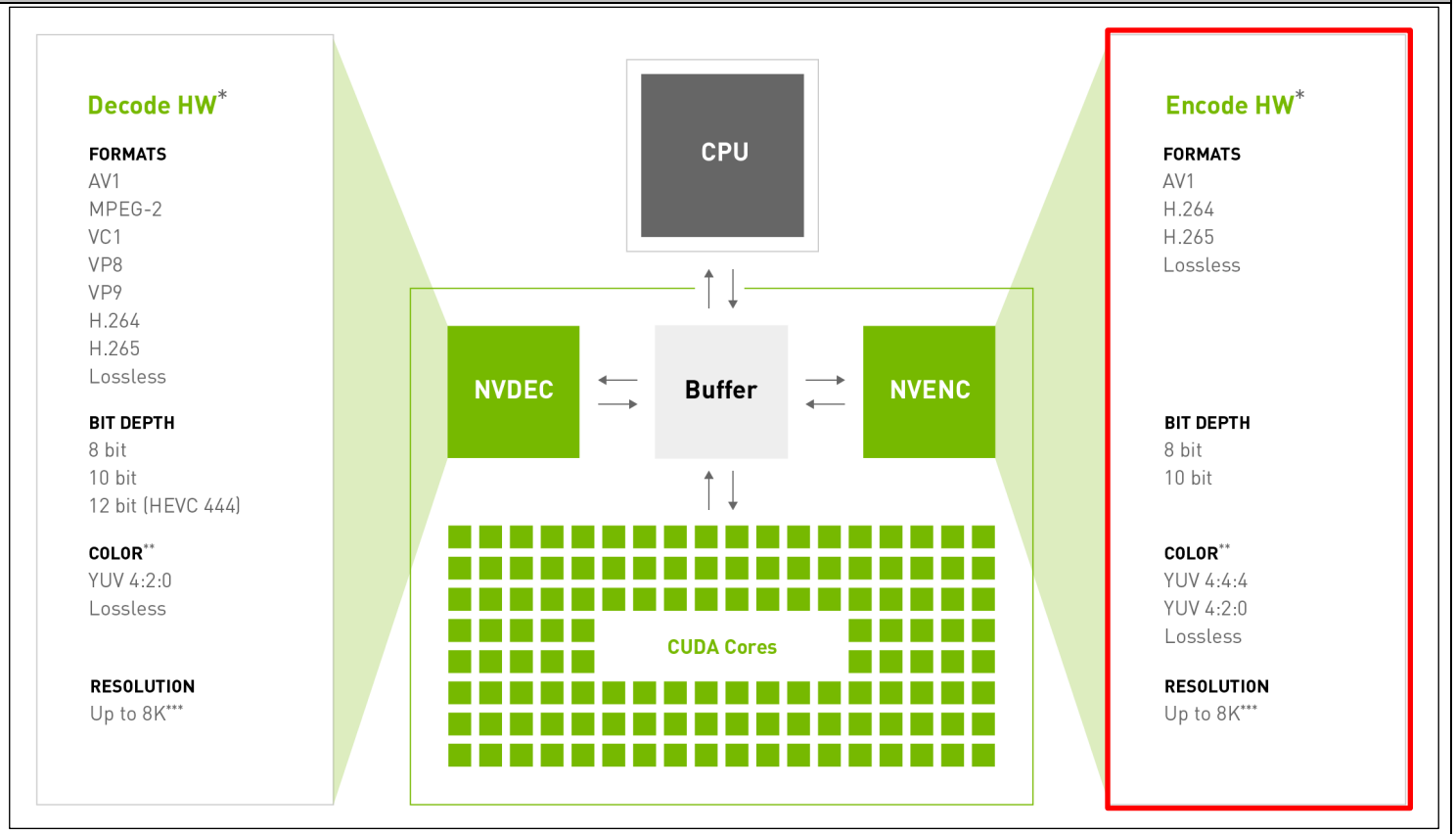
https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

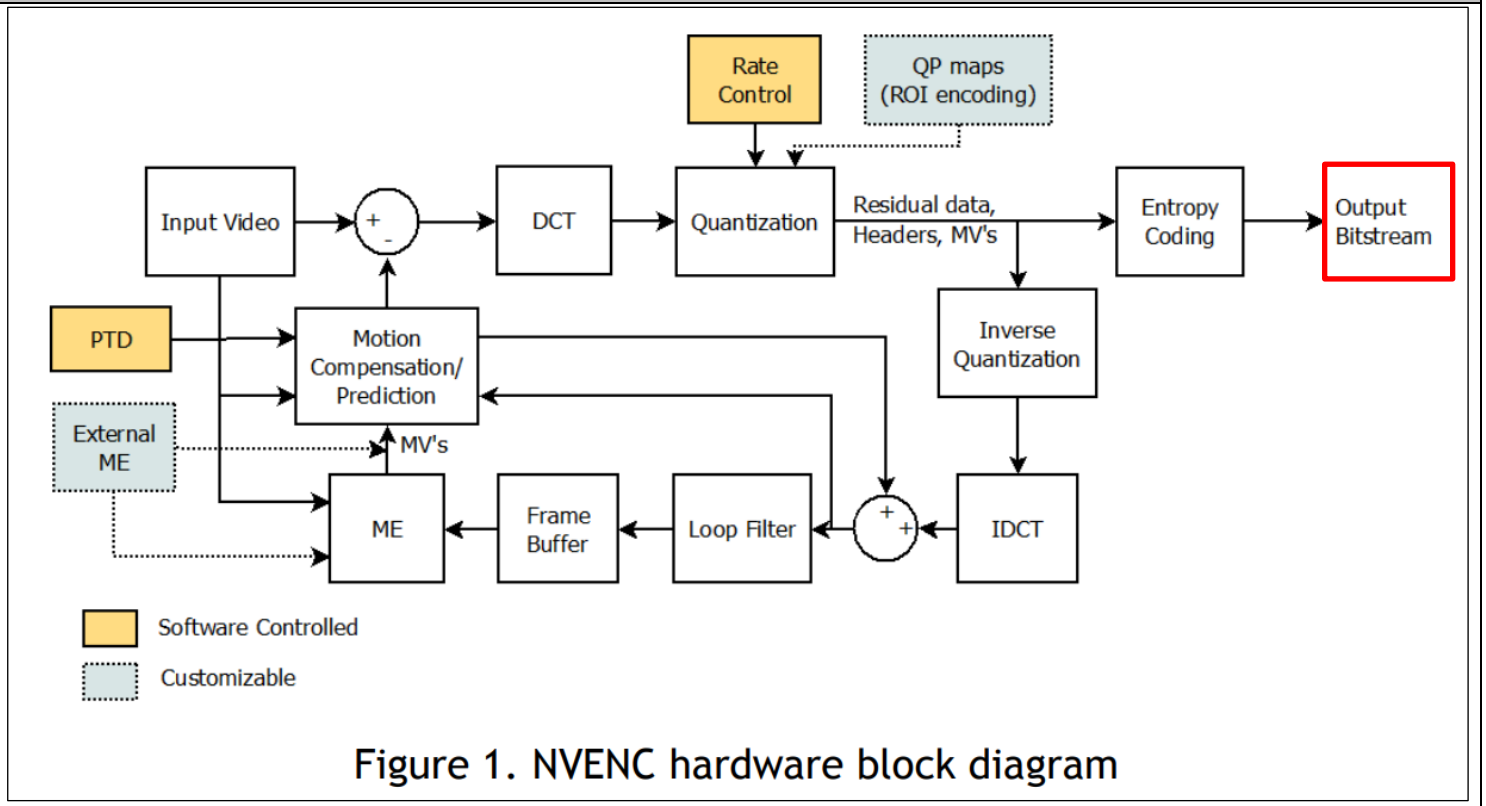


https://www.nvidia.com/en-us/geforce/graphics-cards/compare/

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://developer.nvidia.com/sites/default/files/akamai/designworks/images-capture/nv-proviz-encode-decode-diagram-2461893.png

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Figure 1. NVENC hardware block diagram https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4A] setting an emphasis level to	Samsung or users of the Samsung Accused Products perform: setting an emphasis level to the input video signal in accordance with at least one factor among control data carried by the input video signal, a picture state

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the input video signal in accordance with at least one factor among control data carried by the input video signal, a picture state detected from the input video signal and encoding conditions for the input video signal;	detected from the input video signal and encoding conditions for the input video signal. Samsung directly infringes and induces infringement by causing a user to perform the Emphasis Map feature of the Accused Products. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasis-level setter (*i.e.*, logical unit controlled by NVENC API for defining a QP (quantization parameter) delta map ("Emphasis Map")) for setting an emphasis level (*i.e.*, QP delta values across the map) to the input video signal (*i.e.*, creating custom QP maps for region of interest (ROI) encoding) in accordance with at least one factor among control data carried by the input video signal, a picture state detected from the input video signal and encoding conditions for the input video signal (*e.g.*, QPs derived from the Rate Control algorithm, identification of "high detail"/complex areas).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3. NVENC BLOCK DIAGRAM

Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.

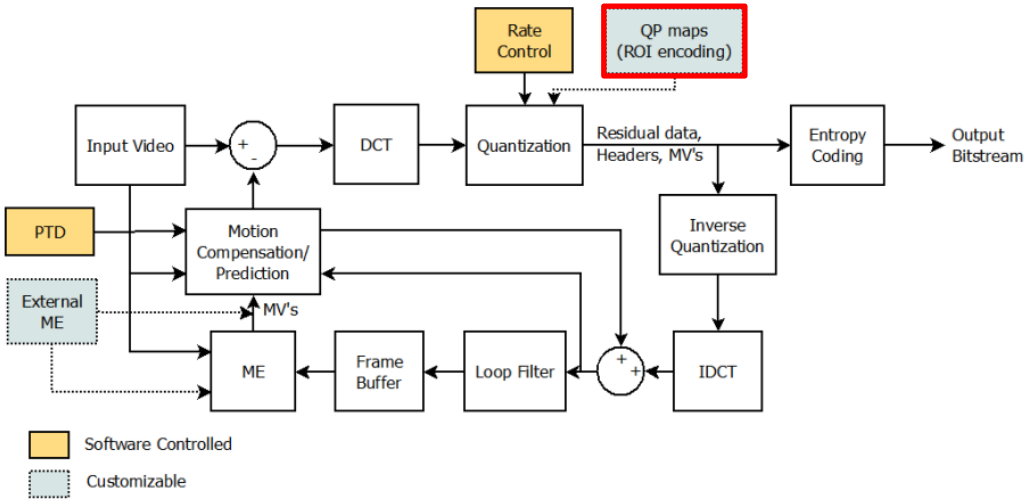


Figure 1. NVENC hardware block diagram

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf at 8

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

EMPHASIS MAP

Region of Interest Encoding

Solution

- Identify "high-detail" areas within the captured image (NVFBC)
- Provide feedback to encoder to treat these areas differently (NVENC)

Generated by NVFBC

5	5	4	5	3	2	1	0
5	5	5	3	3	2	2	0
5	5	4	4	2	1	0	0
3	2	4	3	2	1	1	2
1	1	0	3	2	4	0	0

16 × 16

5 = High detail areas

0 = Low detail areas

Interpreted by NVENC as ΔQP

---	---	--	---	--	-	-	
---	---	---	--	--	-	-	
---	---	--	--	-	-		
--	-	--	--	-	-	-	-
-	-		--	-	--		

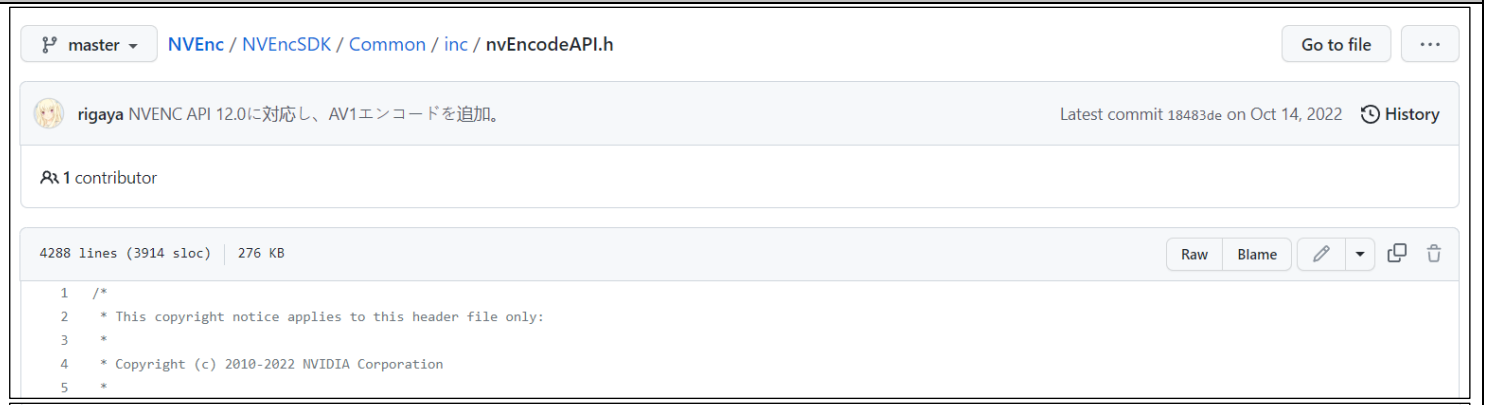
16 × 16

Encoder translates to ΔQP

ΔQP depends on absolute QP

https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
336 /**
337  * QP MAP MODE
338  */
339 typedef enum _NV_ENC_QP_MAP_MODE
340 {
341     NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
342     NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
343     NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
344     NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
345 } NV_ENC_QP_MAP_MODE;
346
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

The NVEncode API Emphasis Map feature specifies regions in the frame to be encoded at different levels of quality (*i.e.*, high-complexity areas at higher quality (at a lower QP)). Subsequently, an adjustment (*i.e.*, emphasis level) is applied to the QP used to encode the respective macroblock. A value of said adjustment depends on an absolute value of the QP as decided by the rate control algorithm and the actual emphasis level for the respective macroblock (*i.e.*, in accordance with at least one factor among control data carried by the input video signal). The picture state is detected from the input video signal and encoding conditions for the input video signal, as the client has "prior knowledge of the image complexity" from the input video signal:

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Multi pass frame encoding

When determining the QP to use for encoding a frame, it is beneficial if NVENC knows the overall complexity of the frame to distribute the available bit budget in the most optimal manner. In some situations, multi-pass encoding may also help catch larger motion between frames. For this purpose, NVENC supports the following types of multi-pass frame encoding modes:

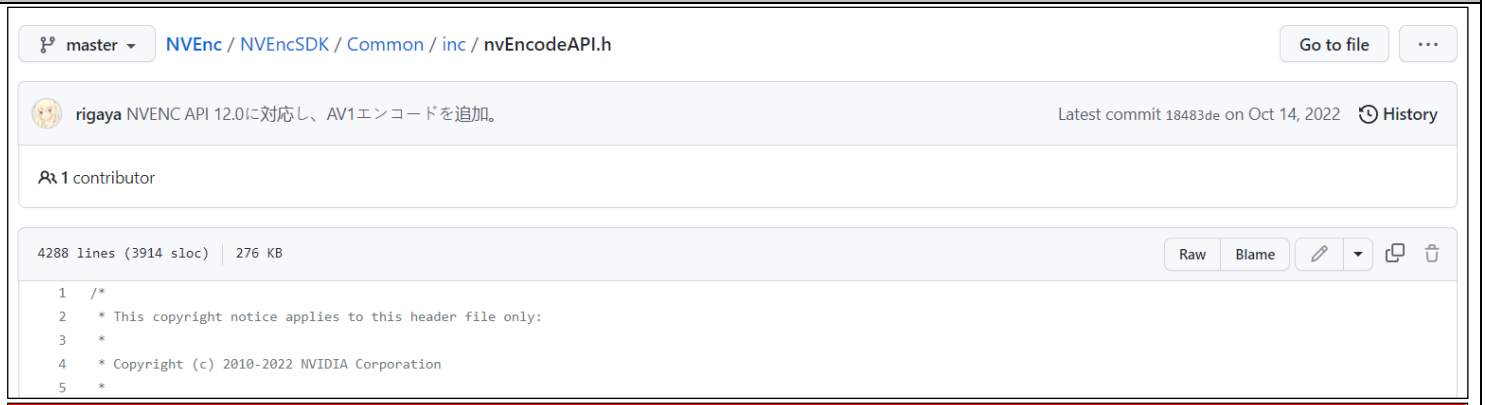
- 1-pass per frame encoding (`NV_ENC_MULTI_PASS_DISABLED`)
- 2-passes per frame, with first pass in quarter resolution and second pass in full resolution (`NV_ENC_TWO_PASS_QUARTER_RESOLUTION`)
- 2-passes per frame, with both passes in full resolution (`NV_ENC_TWO_PASS_FULL_RESOLUION`).

In 1-pass rate control modes, NVENC estimates the required QP for the macroblock and immediately encodes the macroblock. In 2-pass rate control modes, NVENC estimates the complexity of the frame to be encoded and determines bit distribution across the frame in the first pass. In the second pass, NVENC encodes macroblocks in the frame using the distribution determined in the first pass. As a result, with 2-pass rate control modes, NVENC can distribute the bits more optimally within the frame and can reach closer to the target bitrate, especially for CBR encoding. Note, however, that everything else being the same, performance of 2-pass rate control mode is lower than that of 1-pass rate control mode. The client application should choose an appropriate multi-pass rate control mode after evaluating various modes, as each of the modes has its own advantages and disadvantages. `NV_ENC_TWO_PASS_FULL_RESOLUION` generates better statistics for the second pass, whereas `NV_ENC_TWO_PASS_QUARTER_RESOLUTION` results in larger motion vectors being caught and fed as hints to second pass.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

NVEncode API includes a class: "NV_ENC_RC_PARAMS::qpMapMode" which includes Emphasis mode (*i.e.*, NV_ENC_QP_MAP_EMPHASIS) as one of the map modes. In Emphasis mode, "qpDeltaMap" specifies the Emphasis Map Level per macroblock. The map level along with QP for a particular macroblock, selected by rate control, is used to compute a QP modifier. Different Emphasis map levels are associated with different respective Delta QP values. For example, zero Delta QP value is associated with Emphasis Map Level 0. Similarly, very high Delta QP value is associated with Emphasis Map Level 5.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

1 contributor

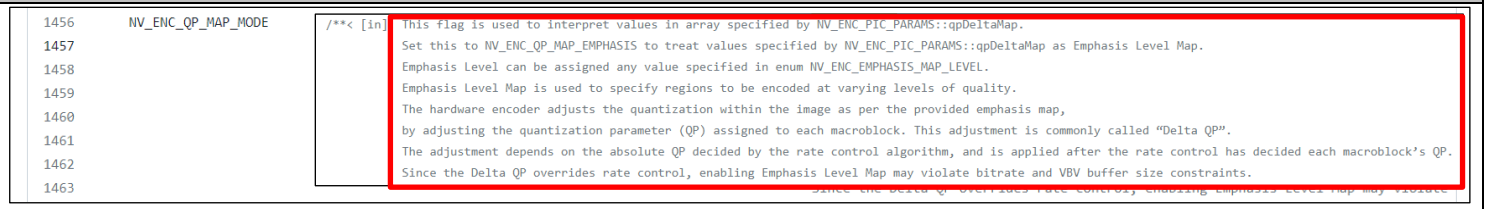
4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
323  /**
324   * Emphasis Levels
325   */
326  typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
327  {
328      NV_ENC_EMPHASIS_MAP_LEVEL_0              = 0x0,       /**< Emphasis Map Level 0, for zero Delta QP value */
329      NV_ENC_EMPHASIS_MAP_LEVEL_1              = 0x1,       /**< Emphasis Map Level 1, for very low Delta QP value */
330      NV_ENC_EMPHASIS_MAP_LEVEL_2              = 0x2,       /**< Emphasis Map Level 2, for low Delta QP value */
331      NV_ENC_EMPHASIS_MAP_LEVEL_3              = 0x3,       /**< Emphasis Map Level 3, for medium Delta QP value */
332      NV_ENC_EMPHASIS_MAP_LEVEL_4              = 0x4,       /**< Emphasis Map Level 4, for high Delta QP value */
333      NV_ENC_EMPHASIS_MAP_LEVEL_5              = 0x5        /**< Emphasis Map Level 5, for very high Delta QP value */
334  } NV_ENC_EMPHASIS_MAP_LEVEL;
335
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

```
1416   typedef struct _NV_ENC_RC_PARAMS
1417   {
1418      uint32_t                     version;
1419      NV_ENC_PARAMS_RC_MODE        rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4B] applying the emphasis processing to the input video signal at the emphasis level to obtain a emphasized video signal;	Samsung or users of the Samsung Accused Products perform: applying the emphasis processing to the input video signal at the emphasis level to obtain an emphasized video signal. Samsung directly infringes and induces infringement by causing a user to perform the Emphasis Map feature of the Accused Products. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises an emphasizer (*i.e.*, NVENC encoder) for applying the emphasis processing (*i.e.*, supports the hardware function of region-of-interest encoding) to the input video signal at the emphasis level (*i.e.*, adjusted QP values, Delta QP) to obtain an emphasized (*e.g.*, quantized) video signal.

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3. NVENC BLOCK DIAGRAM

Apart from the rate control and picture type decision, NVENC can perform all tasks that are a critical part of the end-to-end H.264 and H.265 encoding. The rate control algorithm is implemented in GPU's firmware and controlled via the driver. From the application's perspective, rate control is a hardware function controlled via the parameters exposed in the NVENC APIs. The hardware also provides capability to use external motion estimation engine and custom quantization parameter maps (for ROI "region of interest" encoding). The region of interest encoding has been made available using the "QP delta map" where in the Quantization parameters derived from the Rate Control algorithm can be tweaked using the QP delta map.

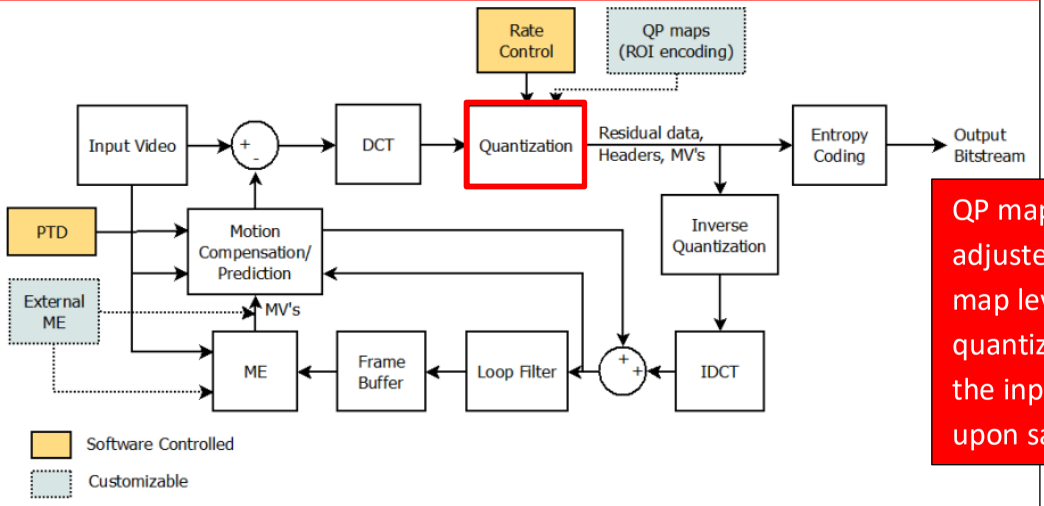


Figure 1. NVENC hardware block diagram

QP maps, including QP values adjusted with respective emphasis map levels, are provided to the quantization block which quantizes the input from the DCT block based upon said adjusted QP values.

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

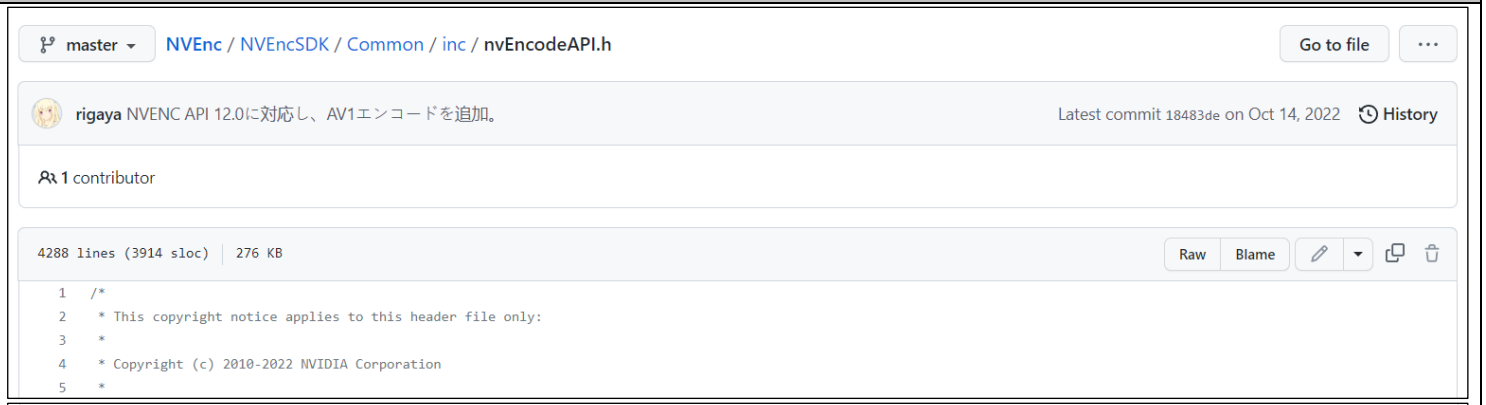
Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ···

rigaya NVENC API 12.0に対応し、AV1エンコードを追加。 Latest commit 18483de on Oct 14, 2022 History

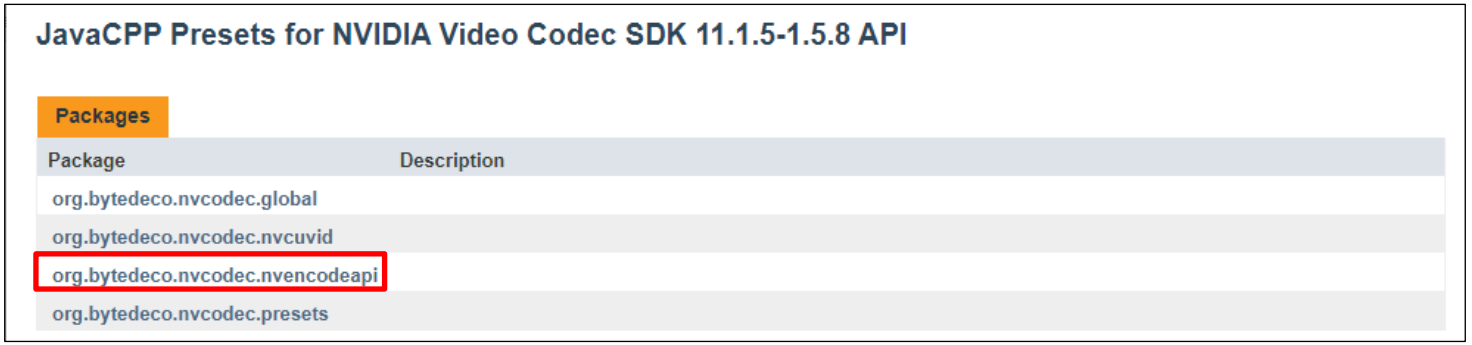
1 contributor

4288 lines (3914 sloc) | 276 KB Raw Blame

```
1   /*
2    * This copyright notice applies to this header file only:
3    *
4    * Copyright (c) 2010-2022 NVIDIA Corporation
5    *
```

```
336  /**
337   * QP MAP MODE
338   */
339  typedef enum _NV_ENC_QP_MAP_MODE
340  {
341      NV_ENC_QP_MAP_DISABLED              = 0x0,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
342      NV_ENC_QP_MAP_EMPHASIS              = 0x1,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
343      NV_ENC_QP_MAP_DELTA                 = 0x2,      /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
344      NV_ENC_QP_MAP                       = 0x3,      /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
345  } NV_ENC_QP_MAP_MODE;
346
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h



JavaCPP Presets for NVIDIA Video Codec SDK 11.1.5-1.5.8 API

Packages

Package	Description
org.bytedeco.nvcodec.global	
org.bytedeco.nvcodec.nvcuvid	
org.bytedeco.nvcodec.nvencodeapi	
org.bytedeco.nvcodec.presets	

http://bytedeco.org/javacpp-presets/nvcodec/apidocs/overview-summary.html

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

org.bytedeco.nvcodec.nvencodeapi

Class NV_ENC_PIC_PARAMS

...

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description

...

qpDeltaMap

```
@Cast(value="int8_t*")
public BytePointer qpDeltaMap()
```

[in]: Specifies the pointer to signed byte array containing value per MB for H264 and per CTB for HEVC in raster scan order for the current picture, which will be interpreted depending on NV_ENC_RC_PARAMS::qpMapMode. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_DELTA, qpDeltaMap specifies QP modifier per MB for H264 and per CTB for HEVC. This QP modifier will be applied on top of the QP chosen by rate control. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_EMPHASIS, qpDeltaMap specifies Emphasis Level Map per MB for H264. This level value along with QP chosen by rate control is used to compute the QP modifier, which in turn is applied on top of QP chosen by rate control. If NV_ENC_RC_PARAMS::qpMapMode is NV_ENC_QP_MAP_DISABLED, value in qpDeltaMap will be ignored.

http://bytedeco.org/javacpp-presets/nvcodec/apidocs/org/bytedeco/nvcodec/nvencodeapi/NV_ENC_PIC_PARAMS.html

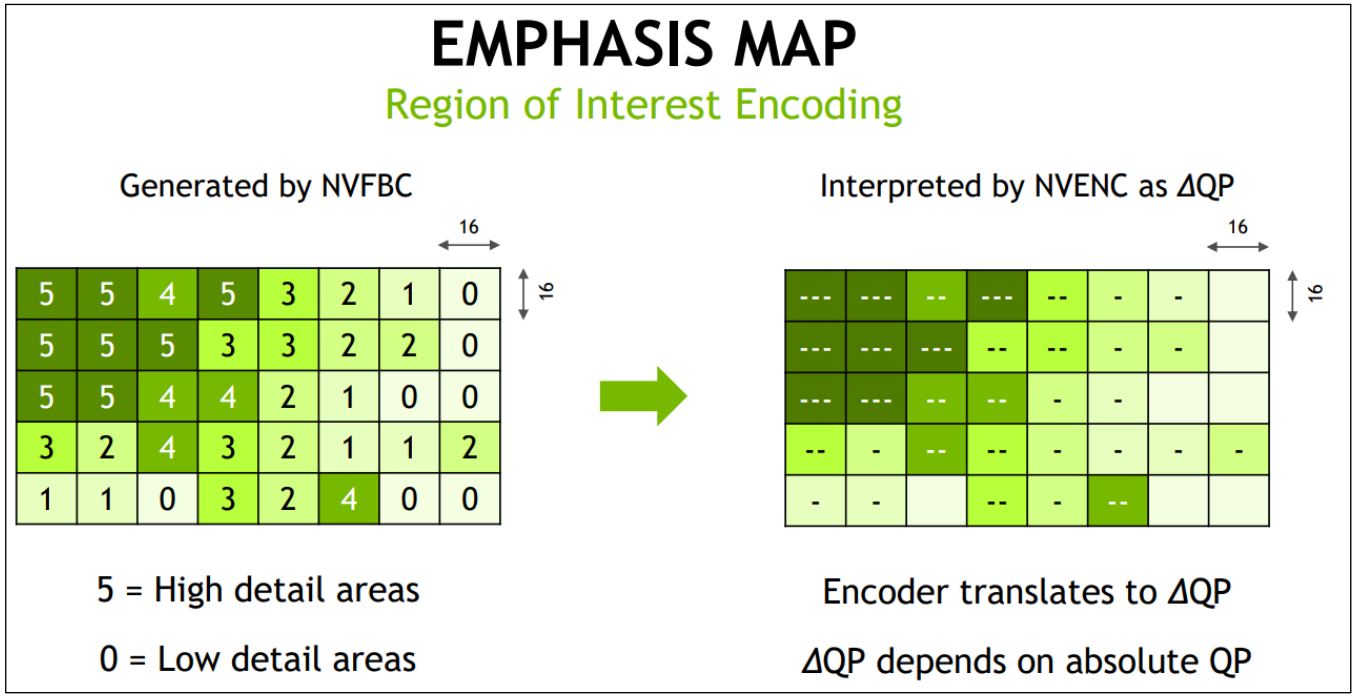
```
1416   typedef struct _NV_ENC_RC_PARAMS
1417   {
1418      uint32_t                    version;
1419      NV_ENC_PARAMS_RC_MODE       rateControlMode;                    /**< [in]: Specifies the rate control mode. Check support for various rate control modes using
```

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

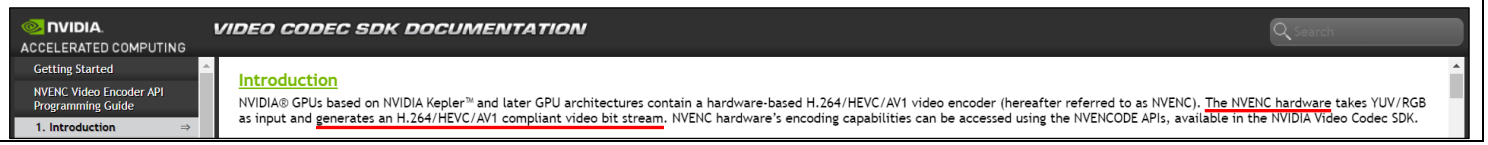
```
1456        NV_ENC_QP_MAP_MODE      /**< [in]
1457
1458
1459
1460
1461
1462
1463
```

This flag is used to interpret values in array specified by NV_ENC_PIC_PARAMS::qpDeltaMap.
Set this to NV_ENC_QP_MAP_EMPHASIS to treat values specified by NV_ENC_PIC_PARAMS::qpDeltaMap as Emphasis Level Map.
Emphasis Level can be assigned any value specified in enum NV_ENC_EMPHASIS_MAP_LEVEL.
Emphasis Level Map is used to specify regions to be encoded at varying levels of quality.
The hardware encoder adjusts the quantization within the image as per the provided emphasis map,
by adjusting the quantization parameter (QP) assigned to each macroblock. This adjustment is commonly called "Delta QP".
The adjustment depends on the absolute QP decided by the rate control algorithm, and is applied after the rate control has decided each macroblock's QP.
Since the Delta QP overrides rate control, enabling Emphasis Level Map may violate bitrate and VBV buffer size constraints.

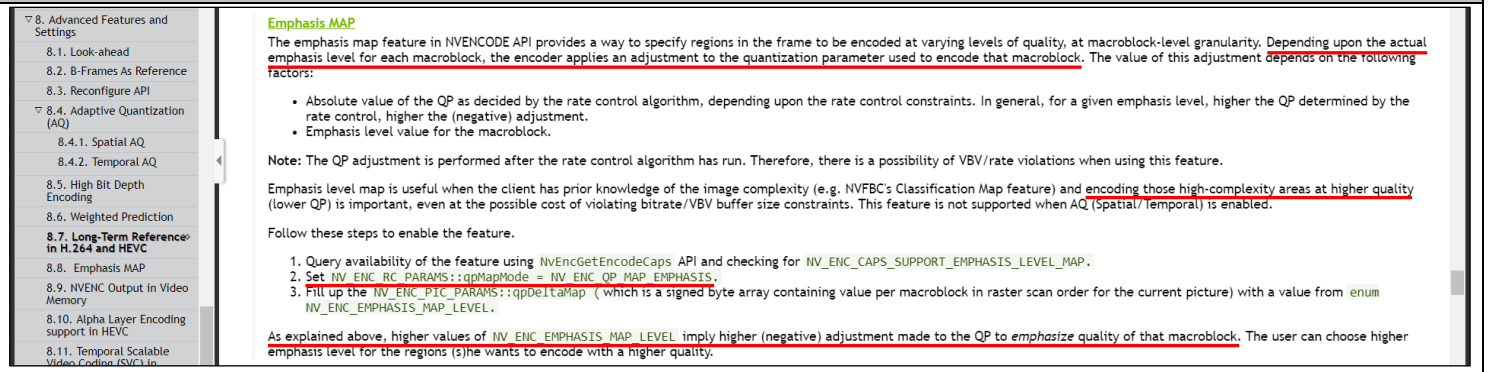
https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h



https://on-demand.gputechconf.com/gtc/2018/presentation/s8601-nvidia-gpu-video-technologies.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4C] encoding the emphasized video signal to obtain a video bitstream; and	Samsung or users of the Samsung Accused Products perform: encoding the emphasized video signal to obtain a video bitstream. Samsung directly infringes and induces infringement by causing a user to perform the Emphasis Map feature of the Accused Products. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises a video-emphasis encoding apparatus for applying an emphasis processing to an input video signal to obtain a video bitstream. NVENC encoder, based on the information of the regions specified by NVEncode API, applies an adjustment to the quantization parameter used to encode a macroblock for obtaining a video bitstream, depending on an emphasis level for the respective macroblock (*i.e.*, an encoder for encoding the emphasized video signal to obtain a video bitstream). For example, higher values of emphasis map level would imply higher adjustment made to the QP to emphasize the quality of the respective macroblock. 

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

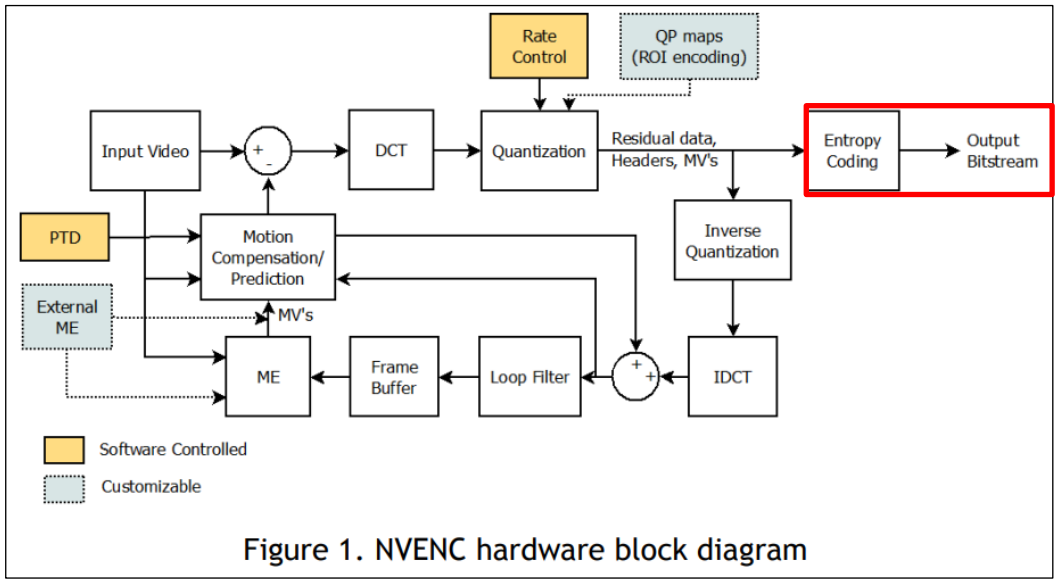


Figure 1. NVENC hardware block diagram

https://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4D] multiplexing the video bitstream and data on the emphasis level.	Samsung or users of the Samsung Accused Products perform: multiplexing the video bitstream and data on the emphasis level. Samsung directly infringes and induces infringement by causing a user to perform the Emphasis Map feature of the Accused Products. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy Book Odyssey laptop with the NVIDIA RTX 3050Ti Max-Q comprises a multiplexer for multiplexing the video bitstream and data on the emphasis level (*e.g.*, *mb_qp_delta*).

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Emphasis MAP

The emphasis map feature in NVENCODE API provides a way to specify regions in the frame to be encoded at varying levels of quality, at macroblock-level granularity. Depending upon the actual emphasis level for each macroblock, the encoder applies an adjustment to the quantization parameter used to encode that macroblock. The value of this adjustment depends on the following factors:

- Absolute value of the QP as decided by the rate control algorithm, depending upon the rate control constraints. In general, for a given emphasis level, higher the QP determined by the rate control, higher the (negative) adjustment.
- Emphasis level value for the macroblock.

Note: The QP adjustment is performed after the rate control algorithm has run. Therefore, there is a possibility of VBV/rate violations when using this feature.

Emphasis level map is useful when the client has prior knowledge of the image complexity (e.g. NVFBC's Classification Map feature) and encoding those high-complexity areas at higher quality (lower QP) is important, even at the possible cost of violating bitrate/VBV buffer size constraints. This feature is not supported when AQ (Spatial/Temporal) is enabled.

Follow these steps to enable the feature.

1. Query availability of the feature using `NvEncGetEncodeCaps` API and checking for `NV_ENC_CAPS_SUPPORT_EMPHASIS_LEVEL_MAP.`
2. Set `NV_ENC_RC_PARAMS::qpMapMode = NV_ENC_QP_MAP_EMPHASIS.`
3. Fill up the `NV_ENC_PIC_PARAMS::qpDeltaMap (` which is a signed byte array containing value per macroblock in raster scan order for the current picture) with a value from `enum NV_ENC_EMPHASIS_MAP_LEVEL.`

As explained above, higher values of `NV_ENC_EMPHASIS_MAP_LEVEL` imply higher (negative) adjustment made to the QP to *emphasize* quality of that macroblock. The user can choose higher emphasis level for the regions (s)he wants to encode with a higher quality.

https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-video-encoder-api-prog-guide/#emphasis-map

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

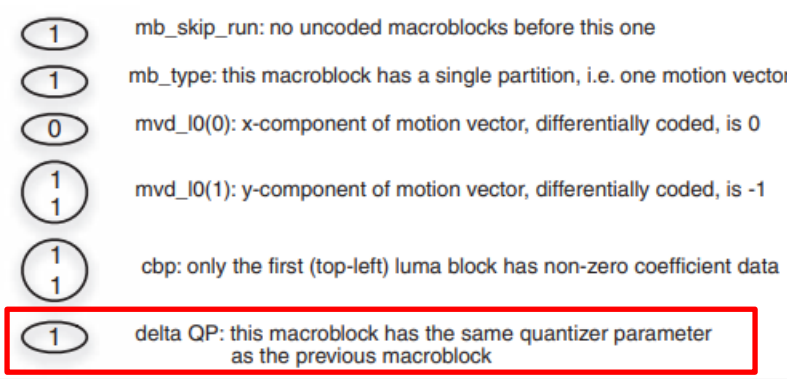
master ▾ NVEnc / NVEncSDK / Common / inc / nvEncodeAPI.h Go to file ⋯

```
/**
 * Emphasis Levels
 */
typedef enum _NV_ENC_EMPHASIS_MAP_LEVEL
{
    NV_ENC_EMPHASIS_MAP_LEVEL_0             = 0x0,      /**< Emphasis Map Level 0, for zero Delta QP value */
    NV_ENC_EMPHASIS_MAP_LEVEL_1             = 0x1,      /**< Emphasis Map Level 1, for very low Delta QP value */
    NV_ENC_EMPHASIS_MAP_LEVEL_2             = 0x2,      /**< Emphasis Map Level 2, for low Delta QP value */
    NV_ENC_EMPHASIS_MAP_LEVEL_3             = 0x3,      /**< Emphasis Map Level 3, for medium Delta QP value */
    NV_ENC_EMPHASIS_MAP_LEVEL_4             = 0x4,      /**< Emphasis Map Level 4, for high Delta QP value */
    NV_ENC_EMPHASIS_MAP_LEVEL_5             = 0x5       /**< Emphasis Map Level 5, for very high Delta QP value */
} NV_ENC_EMPHASIS_MAP_LEVEL;

/**
 * QP MAP MODE
 */
typedef enum _NV_ENC_QP_MAP_MODE
{
    NV_ENC_QP_MAP_DISABLED              = 0x0,          /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap have no effect. */
    NV_ENC_QP_MAP_EMPHASIS              = 0x1,          /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as Emphasis level. Currently this is only supported for
    NV_ENC_QP_MAP_DELTA                 = 0x2,          /**< Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP delta map. */
    NV_ENC_QP_MAP                       = 0x3,          /**< Currently This is not supported. Value in NV_ENC_PIC_PARAMS::qpDeltaMap will be treated as QP value.   */
} NV_ENC_QP_MAP_MODE;
```

https://github.com/rigaya/NVEnc/blob/master/NVEncSDK/Common/inc/nvEncodeAPI.h

4.6 The H.264 Syntax

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Example An H.264 decoder receives the following series of bits, describing an inter-coded macroblock or P MB coded using context-adaptive VLCs: 11011111000111000111 0010 (alternate syntax elements highlighted). Figure 4.20 shows the syntax elements in this sequence of bits. The header elements, the first eight bits and six syntax elements, indicate the macroblock type (P, one motion vector),  https://last.hit.bme.hu/download/vidtech/k%C3%B6nyvek/Iain%20E.%20Richardson%20-%20H264%20(2nd%20edition).pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products

3.5 Entropy coder

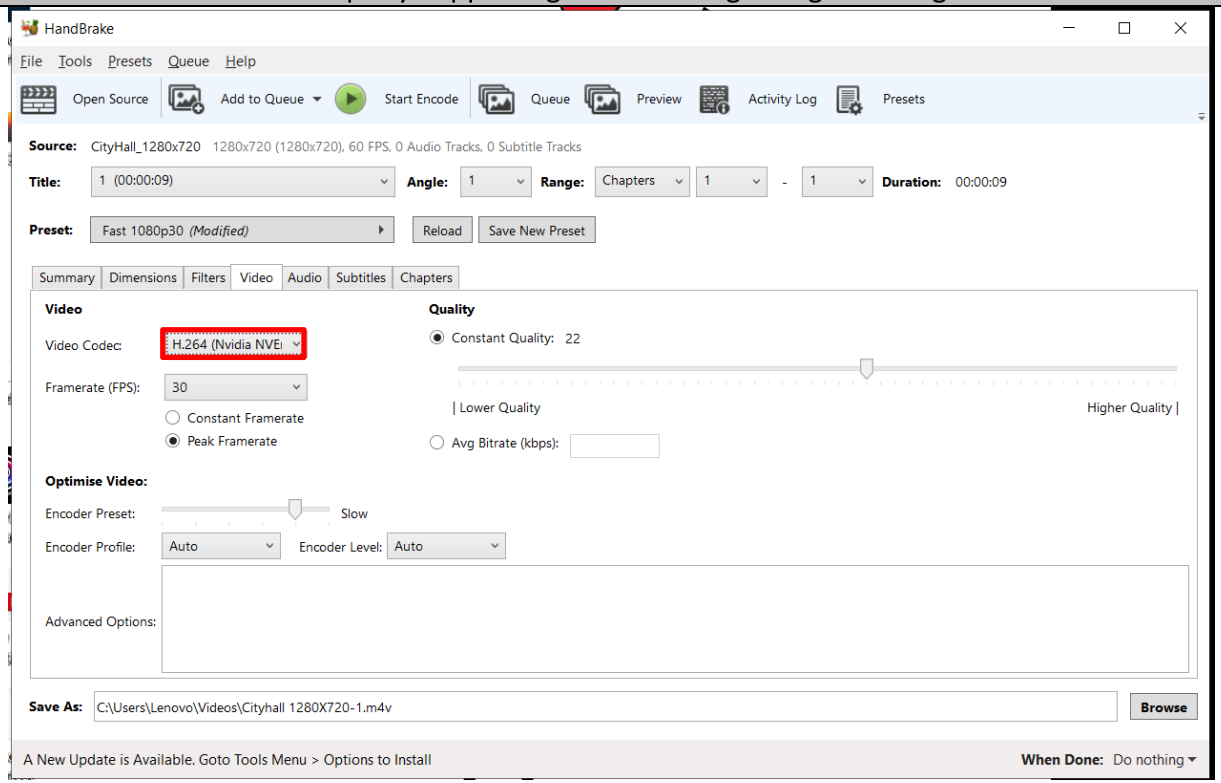
The entropy encoder converts a series of symbols representing elements of the video sequence into a compressed bitstream suitable for transmission or storage. Input symbols may include quantized transform coefficients, run-level or zerotree encoded as described in section 3.4.4, motion vectors with integer or sub-pixel resolution, marker codes that indicate a resynchronization point in the sequence, macroblock headers, picture headers, sequence headers etc and supplementary information, 'side' information that is not essential for correct decoding.

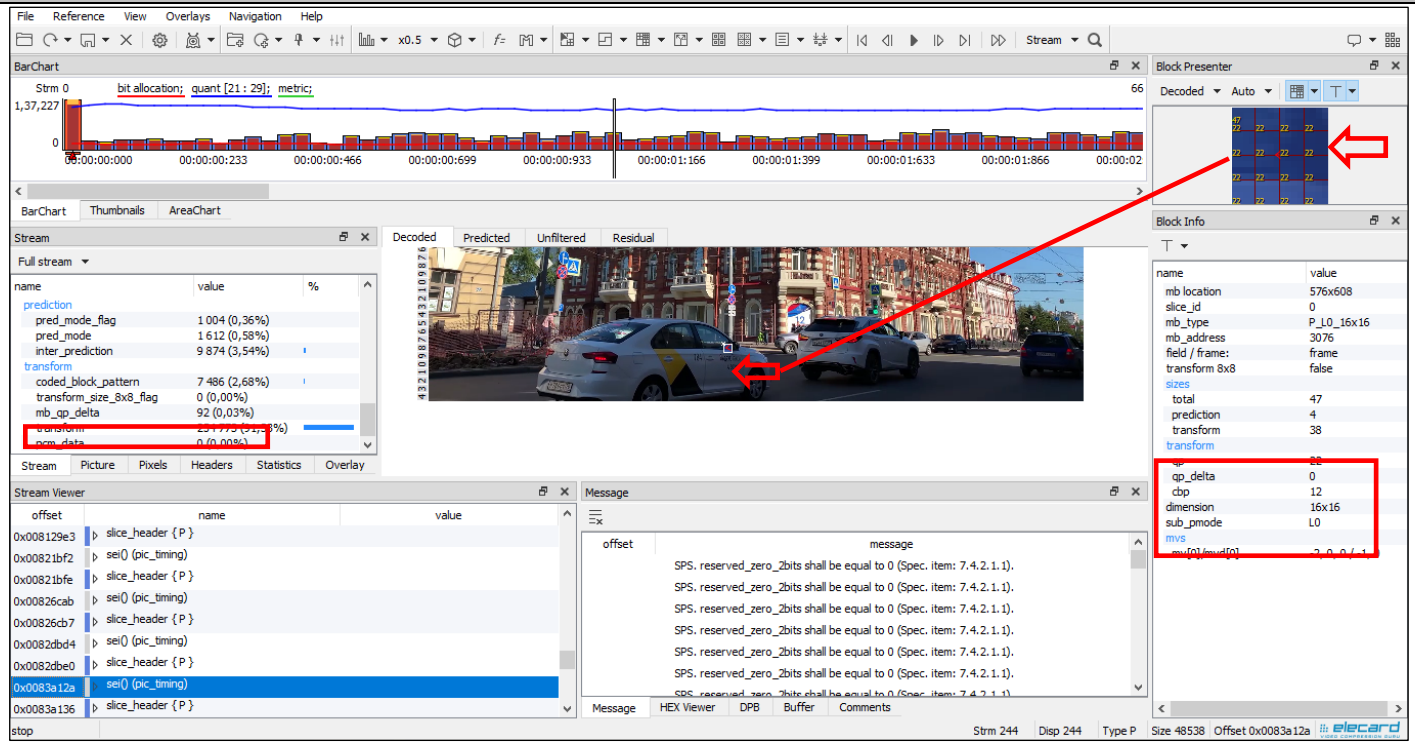
3.5.1 Predictive coding

...

The quantization parameter or quantizer step size controls the trade-off between compression efficiency and image quality. In a real-time video CODEC it may be necessary to modify the quantization within an encoded frame, for example to change the compression ratio in order to match the coded bit rate to a transmission channel rate. It is usually sufficient to change the parameter by a small amount between successive coded macroblocks. The modified quantization parameter must be signalled to the decoder and instead of sending a new quantization parameter value, it may be preferable to send a delta or difference value, e.g. +/−1 or +/−2, indicating the change required. Fewer bits are required to encode a small delta value than to encode a completely new quantization parameter.

https://last.hit.bme.hu/download/vidtech/k%C3%B6nyvek/Iain%20E.%20Richardson%20-%20H264%20(2nd%20edition).pdf

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Internal Testing

Claim – 6,845,128	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Internal Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claims 6-8 and 10 (the "Asserted Claims") of U.S. Patent No. 8,090,025 (the "'025 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung 8K Neo QLED QN800A television, depicted in this chart, is believed to be representative of other televisions and display products, including at least the AU7000, AU8000, AU9000, BU8000, BU8500, Q60A, Q60B, Q60C, Q700T, Q70A, Q70B, Q70C, Q800T, Q80A, Q80B, Q80C, Q900TS, Q90T, Q950TS, Q95T, QN700A, QN700B, QN700C, QN800A, QN800B, QN800C, QN85A, QN85B, QN85C, QN900A, QN900B, QN900C, QN90A, QN90B, QN91A, QN95A, QN95B, QN95C, S90C, S95B, S95C, the Frame 2021, the Frame 2022, The Serif 2022, Q95T, Q90T, Q80T, Q70T, Q60T, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000, LS01T, LS03A, LS03T, LS05T, LST7, QP75A-8K, LSP9T, HBU8000, HQ60B, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra, Galaxy Book Pro, Galaxy Book Pro 360, Galaxy Book2 Pro, Galaxy Book2 Pro 360, Galaxy Book2 Business 14', Galaxy Book Odyssey, Galaxy Book3 Ultra, Galaxy Book3 Pro, Galaxy Book3 Pro 360, and Galaxy Book3 360 models for the purposes of infringement of the '025 Patent. Based on publicly available information, ACT believes that the Samsung 8K Neo QLED QN800A television, charted herein, and other television/display products share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the listed televisions/display products all purport to support the AV1 codec. Should discovery yield documents or information demonstrating that Defendants sell other infringing products which are not represented by the Samsung 8K Neo QLED QN800A television, ACT reserves its right to amend its infringement contentions to include a chart mapping the claims of the '025 Patent to the structure and function of the newly-discovered products.

Samsung directly infringes each of the Asserted Claims by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

Samsung indirectly infringes the Asserted Claims in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through their operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claim; and

(ii) marketing and advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers, including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claims, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claims but has taken deliberate actions to avoid learning these facts.

Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

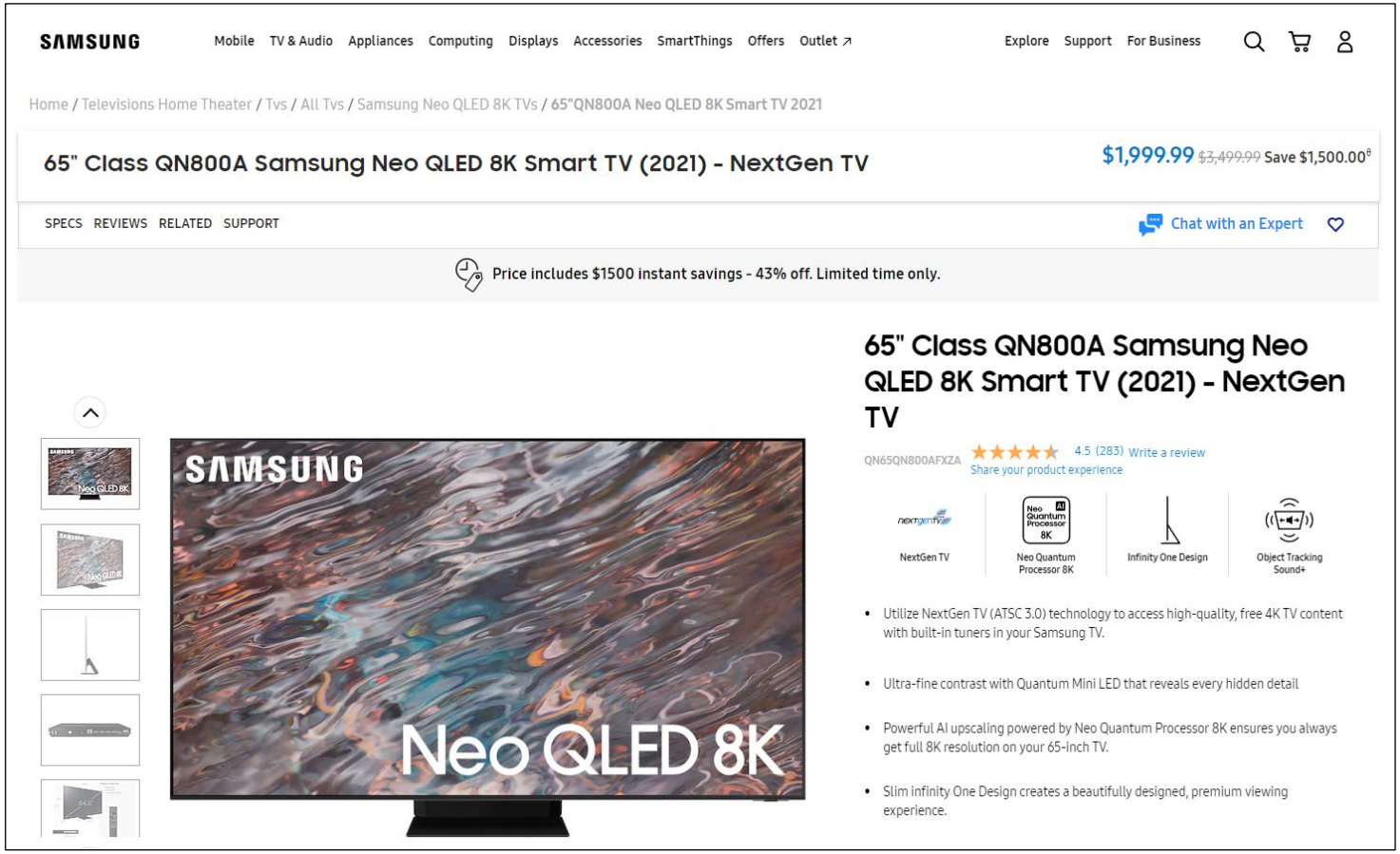
On information and belief, the charted version of Samsung's products is representative of all versions of the Accused Products, including all variants of the Accused Products made, sold, offered for sale, or used on any operating system.

ACT does not concede that any claims of the '025 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

Claim – 8,090,025	Exemplary Supporting Evidence Regarding Samsung's Accused Products

[6P] A moving-picture decoding apparatus comprising:	The Samsung Accused Products are each: A moving-picture decoding apparatus. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A is a moving-picture decoding apparatus that implements moving picture decoding using the AV1 codec.



https://www.samsung.com/us/televisions-home-theater/tvs/samsung-neo-qled-8k/65-qn800a-neo-qled-8k-smart-tv-2021-qn65qn800afxza/

Compatible Samsung TVs

Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec.

TV Series	Models
Neo QLED 8K	QN900A, QN800A, QN700A
QLED 8K	Q950T, Q900T, Q800T, Q700T
Neo QLED 4K	QN95A, QN91A, QN90A, QN85A
QLED 4K	Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T
Crystal UHD	AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000
The Serif	LS01T
The Frame	LS03A, LS03T

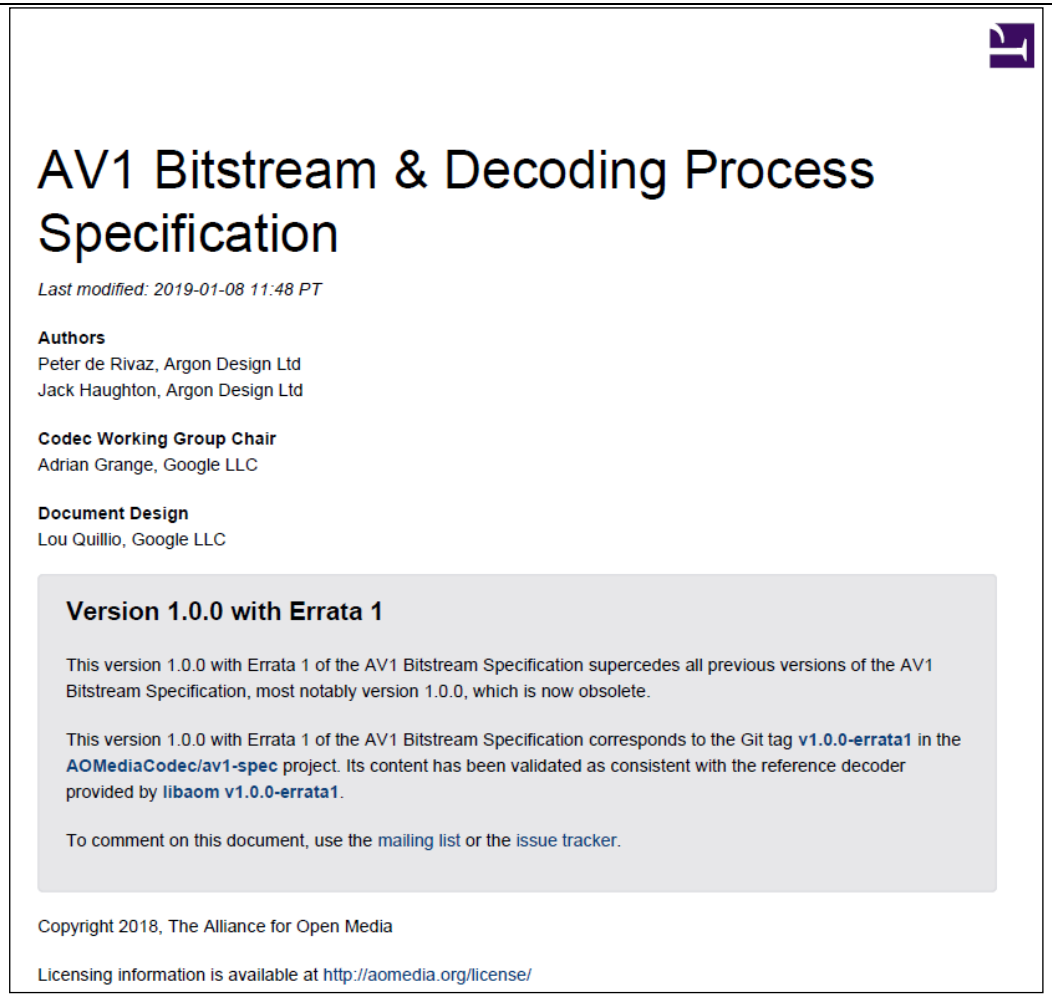
https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

The AV1 codec is used for video decoding, which is moving picture decoding.



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

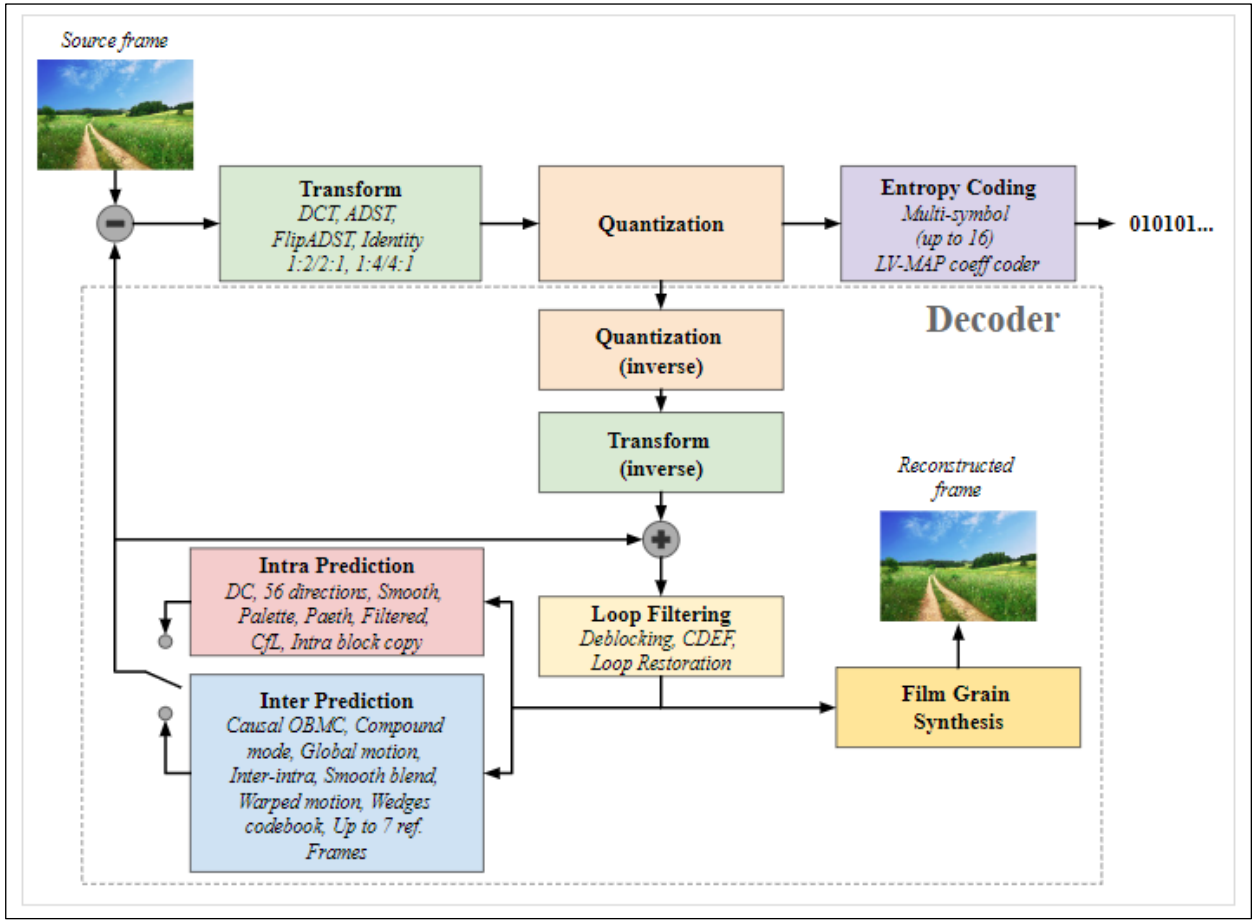
To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

The AV1 decoder is a moving-picture coding apparatus that implements a moving-picture decoding method, as described below to reproduce a reconstructed frame.



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6A1] a demultiplexer to demultiplex coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding,	The Samsung Accused Products comprise: a demultiplexer to demultiplex coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a demultiplexer that demultiplexes coded data (.mp4, .mkv, or .webm files) from an input signal based on a specific syntax structure (open bitstream unit (OBU) syntax structure). **What is an AV1 file?** An AV1 file is a video encoded with AV1 compression, which is an open-source video codec developed by the Alliance for Open Media. AV1 videos are more commonly saved with the .MP4, .MKV, and .WEBM extension(s). https://fileinfo.com/extension/av1

2022 TV Video Specifications

Samsung Smart TV provides rich capabilities for video playback, using the available audio and video decoders. This topic describes the media formats supported by Samsung Smart TV, and the limitations on their support.

22TV_PREMIUM1 (8K)

File Extension	Container	Video Codec	Resolution	Frame Rate (fps)	Bit Rate (Mbps)	Audio Codec
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS	H.264 BP/MP/HP	4096x2160	60	60	AC3 LPCM ADPCM(IMA, MS) AAC HE-AAC WMA DD+ MPEG(MP3) MPEG-H AC-4 OPUS G.711(A-Law, μ-Law)
			1920x1080	120		
		HEVC (H.265 - Main, Main10)	7680x4320	60	100	
			3840x2160	120		
		Motion JPEG		30	80	
		MPEG-4 SP/ASP	1920x1080	60	20	
		Window Media Video 9 (VC-1)				
		MPEG-2				
		MPEG-1				
		Microsoft MPEG-4 version 1/2/3				
		Window Media Video 7 (WMV1) / Video 8 (WMV2)		30		
		H.263 Sorenson				
		VP6				
		AV1	7680x4320	60	80	
				120		
*.webm	WebM	VP8	1920x1080	60	20	Vorbis
		VP9 (Profile 0/2)	3840x2160		80	
		AV1	7680x4320		60	
*.rmvb	RMVB	RV8/9/10 (RV30/40)	1920x1080		20	RealAudio 6

https://developer.samsung.com/smarttv/develop/specifications/media-specifications/2022-tv-video-specifications.html

II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

We refer to [9] for bit field definitions and more detailed consideration of high-level syntax.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 input signal to Samsung TVs is obtained by multiplexing a coded bitstream obtained by predictive coding (*i.e.*, predictive-transform coding), border motion-vector data (*i.e.*, interframe motion compensation), and post-quantization data (*i.e.*, reconstructed samples based on dequantized coefficients and inverse transforms)) obtained by quantization in the predictive coding.

	The AV1 codec supports input video signals in the 4:0:0 (monochrome), 4:2:0, 4:2:2, and 4:4:4 formats. The allowed pixel representations are 8, 10, and 12 bit. The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D unitary transform to further remove the spatial correlations, and the transform coefficients are quantized. Both the prediction syntax elements and the quantized transform coefficient indexes are then entropy coded using arithmetic coding. There are three optional in-loop postprocessing filter stages to enhance the quality of the reconstructed frame for reference by subsequent coded frames. A normative film grain synthesis unit is also available to improve the perceptual quality of the displayed frames. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 As shown below, the AV1 input signal is obtained by predictive coding (*i.e.*, predictive-transform coding). One example of predictive coding is through inter frame prediction:

V. Coding Block Operations

C. Inter Frame Prediction

Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector.

Overlapped Block Motion Compensation The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

The variable isCompound is set equal to RefFrames[candRow][candCol][1] > INTRA_FRAME.

The prediction arrays are formed by the following ordered steps:

1. The rounding variables derivation process specified in section 7.11.3.2 is invoked with the variable isCompound as input.
2. If plane is equal to 0 and motion_mode is equal to LOCALWARP, the warp estimation process in section 7.11.3.8 is invoked.
3. If plane is equal to 0 and motion_mode is equal to LOCALWARP and LocalValid is equal to 1, the setup shear process specified in section 7.11.3.6 is invoked with LocalWarpParams as input, and the output warpValid is assigned to LocalValid (the other outputs are discarded).
4. The variable refList is set equal to 0.
5. The variable refFrame is set equal to RefFrames[candRow][candCol][refList].

6. If (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV) and GmType[refFrame] > TRANSLATION, the setup shear process specified in section 7.11.3.6 is invoked with gm_params[refFrame] as input, and the output warpValid is assigned to globalValid (the other outputs are discarded).

7. The variable useWarp (a value of 1 indicates local warping, 2 indicates global warping) is derived as follows:

    - If w < 8 or h < 8, useWarp is set equal to 0.
    - Otherwise, if force_integer_mv is equal to 1, useWarp is set equal to 0.
    - Otherwise, if motion_mode is equal to LOCALWARP and LocalValid is equal to 1, useWarp is set equal to 1.
    - Otherwise, if all of the following are true, useWarp is set equal to 2.
        - (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV).
        - GmType[refFrame] > TRANSLATION.
        - is_scaled(refFrame) is equal to 0.
        - globalValid is equal to 1.
    - Otherwise, useWarp is set equal to 0.

8. The motion vector array mv is set equal to Mvs[candRow][candCol][refList].

9. The variable refIdx specifying which reference frame is being used is set as follows:

    - If use_intrabc is equal to 0, refIdx is set equal to ref_frame_idx[refFrame - LAST_FRAME].
    - Otherwise (use_intrabc is equal to 1), refIdx is set equal to -1 and RefFrameWidth[-1] is set equal to FrameWidth, RefFrameHeight[-1] is set equal to FrameHeight, and RefUpscaledWidth[-1] is set equal to UpscaledWidth. (These values ensure that the motion vector scaling has no effect.)

10. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, x, y, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY.

11. If use_intrabc is equal to 1, RefFrameWidth[-1] is set equal to MiCols * MI_SIZE, RefFrameHeight[-1] is set equal to MiRows * MI_SIZE, and RefUpscaledWidth[-1] is set equal to MiCols * MI_SIZE. (These values are needed to avoid intrabc prediction being cropped to the frame boundaries.)

12. If useWarp is not equal to 0, the block warp process in section 7.11.3.5 is invoked with useWarp, plane, refList, x, y, i8, j8, w, h as inputs and the output is merged into the 2D array preds[refList] for i8 = 0..((h-1) >> 3) and for j8 = 0..((w-1) >> 3). (Each invocation fills in a block of output of size w by h at x offset j8 * 8 and y offset i8 * 8.)

13. If useWarp is equal to 0, the block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, w, h, candRow, candCol as inputs and the output is assigned to the 2D array preds[refList].

14. If isCompound is equal to 1, then the variable refList is set equal to 1 and steps 5 to 13 are repeated to form the prediction for the second reference.

An array named Mask is prepared as follows:

- If compound_type is equal to COMPOUND_WEDGE and plane is equal to 0, the wedge mask process in section 7.11.3.11 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_INTRA, the intra mode variant mask process in section 7.11.3.13 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_DIFFWTD and plane is equal to 0, the difference weight mask process in section 7.11.3.12 is invoked with preds, w, h as inputs.
- Otherwise, no mask array is needed.

If compound_type is equal to COMPOUND_DISTANCE, the distance weights process in section 7.11.3.15 is invoked with candRow and candCol as inputs.

The inter predicted samples are then derived as follows:

- If isCompound is equal to 0 and IsInterIntra is equal to 0, CurrFrame[plane][y + i][x + j] is set equal to Clip1(preds[0][i][j]) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_AVERAGE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(preds[0][i][j] + preds[1][i][j], 1 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_DISTANCE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(FwdWeight * preds[0][i][j] + BckWeight * preds[1][i][j], 4 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise, the mask blend process in section 7.11.3.14 is invoked with preds, plane, x, y, w, h as inputs.

If motion_mode is equal to OBMC, the overlapped motion compensation in section 7.11.3.9 is invoked with plane, w, h as inputs.

7.11.3.4. Block inter prediction process

The inputs to this process are:

- a variable plane,
- a variable refIdx specifying which reference frame is being used (or -1 for intra block copy),
- variables x and y giving the block location with in units of 1/1024 th of a sample,
- variables xStep and yStep giving the step size in units of 1/1024 th of a sample,
- variables w and h giving the width and height of the block in units of samples,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The output from this process is the 2D array named pred containing inter predicted samples.

A variable ref specifying the reference frame contents is set as follows:

- If refIdx is equal to -1, ref is set equal to CurrFrame.
- Otherwise (refIdx is greater than or equal to 0), ref is set equal to FrameStore[refIdx].

The variables subX and subY are set equal to the subsampling for the current plane as follows:

- If plane is equal to 0, subX is set equal to 0 and subY is set equal to 0.
- Otherwise, subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The variable lastX is set equal to ((RefUpscaledWidth[refIdx] + subX) >> subX) - 1.

The variable lastY is set equal to ((RefFrameHeight[refIdx] + subY) >> subY) - 1.

(lastX and lastY specify the coordinates of the bottom right sample of the reference plane.)

The variable intermediateHeight specifying the height required for the intermediate array is set equal to (((h - 1) * yStep + (1 << SCALE_SUBPEL_BITS) - 1) >> SCALE_SUBPEL_BITS) + 8.

The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. The fractional parts of the motion vectors determine the filtering process. If the fractional part is zero, then the filtering is equivalent to a straight sample copy.

The filtering is applied as follows:

- The array intermediate is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 1 ]
if ( w <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < intermediateHeight; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = x + xStep * c
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              ref[ plane ] [ Clip3( 0, lastY, (y >> 10) + r - 3 ) ]
                           [ Clip3( 0, lastX, (p >> 10) + t - 3 ) ]
        intermediate[ r ][ c ] = Round2(s, InterRound0)
    }
}
```

- The array pred is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 0 ]
if ( h <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < h; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = (y & 1023) + yStep * r
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              intermediate[ (p >> 10) + t ][ c ]
        pred[ r ][ c ] = Round2(s, InterRound1)
    }
}
```

	https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257-263 As shown below, the AV1 input signal is obtained by border motion-vector data (*i.e.*, interframe motion compensation): V. CODING BLOCK OPERATIONS *C. Inter Frame Prediction* Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

	When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector: 1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs.

The function get_obmc_mask returns a blending mask as follows:

```
get_obmc_mask( length ) {
    if ( length == 2 ) {
        return Obmc_Mask_2
    } else if ( length == 4 ) {
        return Obmc_Mask_4
    } else if ( length == 8 ) {
        return Obmc_Mask_8
    } else if ( length == 16 ) {
        return Obmc_Mask_16
    } else {
        return Obmc_Mask_32
    }
}
```

The blending masks are defined as follows:

```
Obmc_Mask_2[2] = { 45, 64 }

Obmc_Mask_4[4] = { 39, 50, 59, 64 }

Obmc_Mask_8[8] = { 36, 42, 48, 53, 57, 61, 64, 64 }

Obmc_Mask_16[16] = { 34, 37, 40, 43, 46, 49, 52, 54,
                     56, 58, 60, 61, 64, 64, 64, 64 }

Obmc_Mask_32[32] = { 33, 35, 36, 38, 40, 41, 43, 44,
                     45, 47, 48, 50, 51, 52, 53, 55,
                     56, 57, 58, 59, 60, 60, 61, 62,
                     64, 64, 64, 64, 64, 64, 64, 64 }
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-278

As shown below, the AV1 input signal is obtained by post-quantization data (*i.e.*, reconstructed samples based on dequantized coefficients and inverse transforms) obtained by quantization (*i.e.*, using transform coefficients to discretize the signal) in the predictive coding:

Transform coefficient

A scalar value, considered to be in a frequency domain, contained in a transform block.

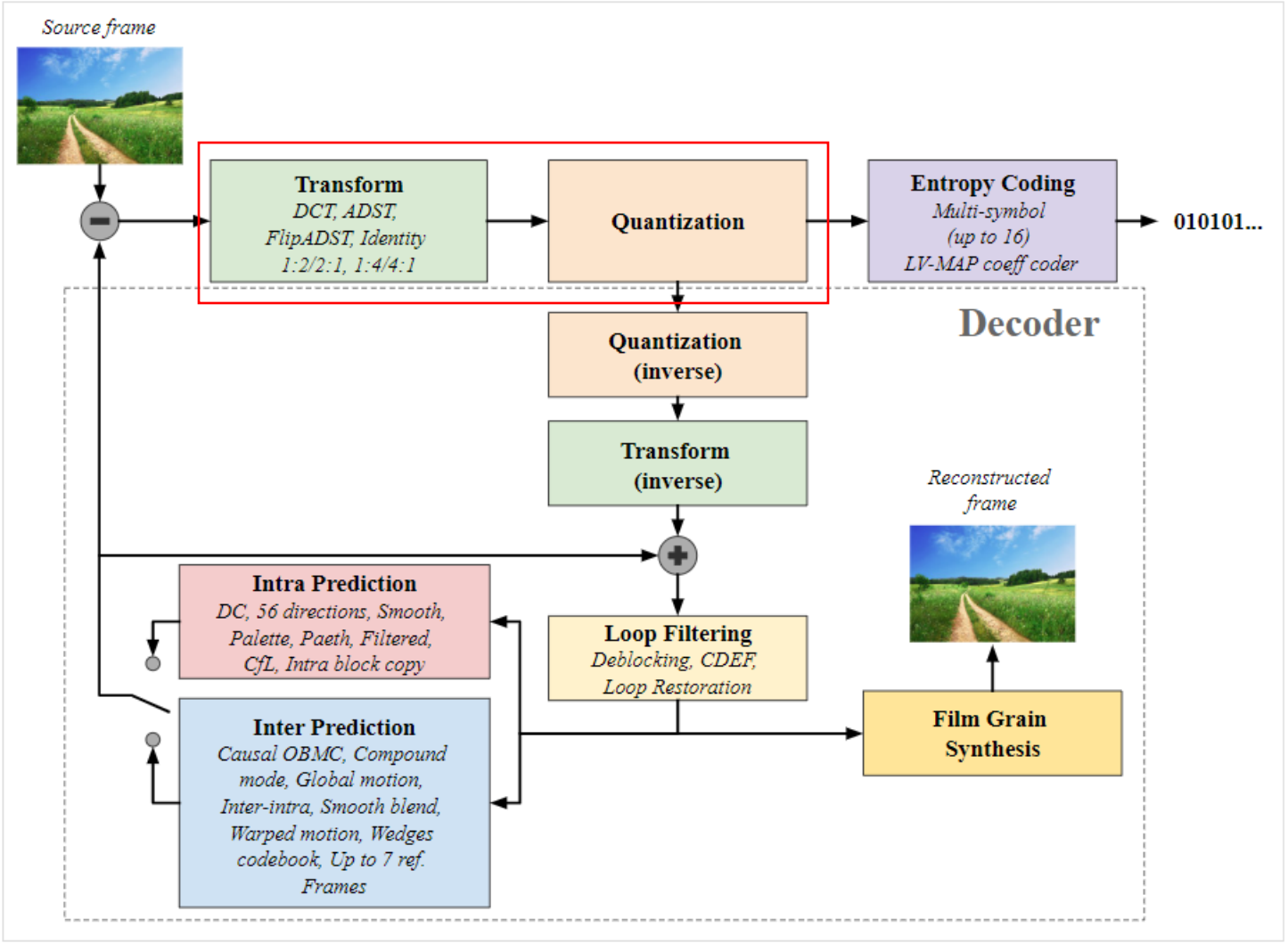
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 6

Quantized coefficient

A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4

	Dequantization The process in which transform coefficients are obtained by scaling the quantized coefficients. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6A2] the coded bitstream obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video	The coded bitstream is obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the coded bitstream input to the Samsung 8K Neo QLED QN800A is obtained by producing and encoding a residual picture that is a residual signal (*i.e.*, prediction residue) between a picture to be coded (*i.e.*, original samples) that is an input moving-picture video signal to be subjected to coding and a predictive picture (*i.e.*, predicted samples) produced from a reference picture (*i.e.*, reference frame).

area of the moving-picture video signal is divided,

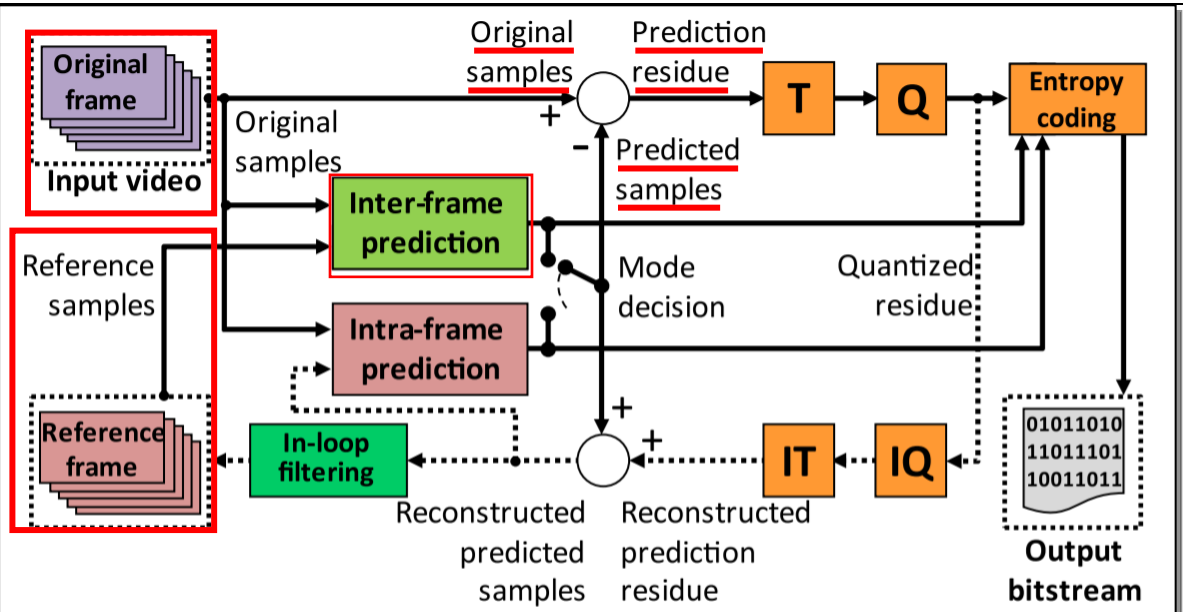


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

The reference picture (*i.e.*, reference frame) is a local decoded video signal (*i.e.*, previously decoded frame) for each of a plurality of rectangular zones (*i.e.*, block). The video area of the moving-picture video signal (*i.e.*, frame) is divided into such rectangular zones (*i.e.*, blocks).

Reference frame
A storage area for a previously decoded frame and associated information.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5

	Frame The representation of video signals in the spatial domain, composed of one luma sample matrix (Y) and two chroma sample matrices (U and V). https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2 **Block** A square or rectangular region of samples. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 1 **Sample** The basic elements that compose the frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5 **Inter coding** Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 3 The plurality of rectangular zones (*i.e.*, several blocks of pixels) are composed of a specific number of pixels (*e.g.*, 128 x 128 pixels, 64 x 64 pixels, 4 x 4 pixels).

V. CODING BLOCK OPERATIONS

A. Coding Block Partitioning

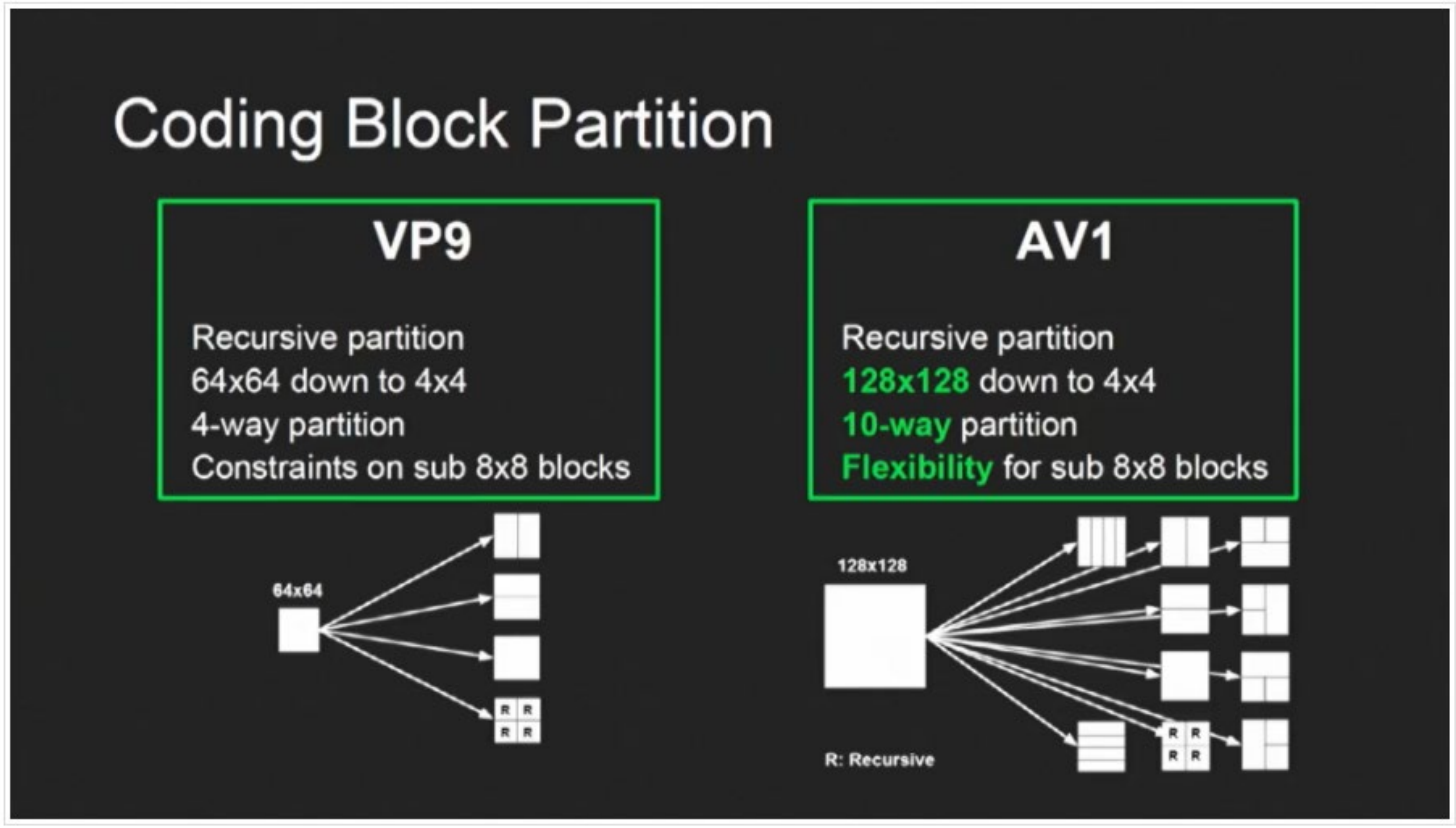
A superblock can be recursively partitioned into smaller block sizes for coding. AV1 inherits the recursive block partitioning design used in VP9. To reduce the overhead cost on prediction mode coding for video signals that are highly correlated, a situation typically seen in 4k videos, AV1 supports a maximum coding block size of 128×128 luma samples. The allowed partition options at each block level include ten possibilities, as shown in Fig. 2. To improve the prediction quality for complex videos, the minimum coding block size is extended to 4×4 luma samples. While such extensions provide more coding flexibility, they have implications for hardware decoders. Certain block size-dependent constraints are specifically designed to circumvent such complications.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D uni-

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	A. BLOCK PARTITIONING Before the prediction process begins, a frame is divided into several blocks of pixels. Each video encoder defines a variable range of block sizes it can use. In AV1, a video frame is partitioned in superblocks (SBs) of size 128×128 or 64×64 pixels. To deliver a locally optimal prediction for each SB, the encoder can further divide each SB using a 10-way partition tree structure, as illustrates Fig. 2. In the figure, filled partitions are final partition modes, but all four partitions of the unfilled partition (SPLIT) can be recursively divided based on the same 10-way tree structure, down to 4×4, which is the smallest supported block size [2]. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

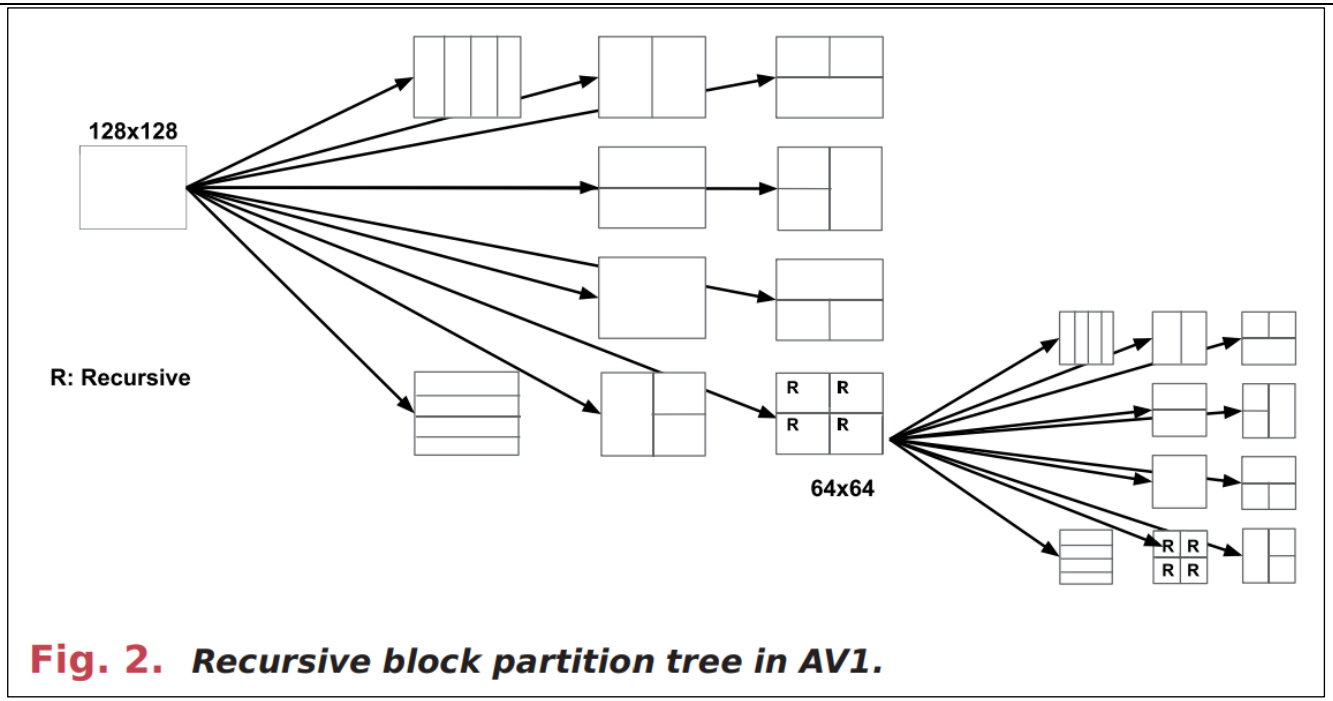


Fig. 2. ***Recursive block partition tree in AV1.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

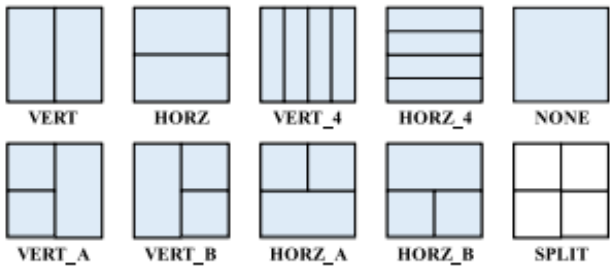


FIGURE 2. AV1 10-way block partition tree structure.

	https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8= Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6A3] obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that	The Samsung Accused Products practice: obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A, through the use of the AV1's block-matching algorithms, such as overlapped block motion compensation (OBMC), obtains the boundary conditions of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones (*i.e.*, "immediate spatial neighbors' motion information to improve the prediction quality for pixels near [a block's] top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides"). After the boundary condition of the block is obtained, the OBMC finds a boundary condition in the reference frame that matches the boundary condition in the current frame to identify the "four reference blocks that have the same reference frame as the current block" and ensure that the boundary condition is met (*i.e.*, "width is the same as the reference block's width, and its height is half of the current block's height").

is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found,	*e) Overlapped block motion compensation:* The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as $$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \quad (15)$$ where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function: $$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \quad (16)$$ where $y = 0, 1, \ldots, H/2-1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

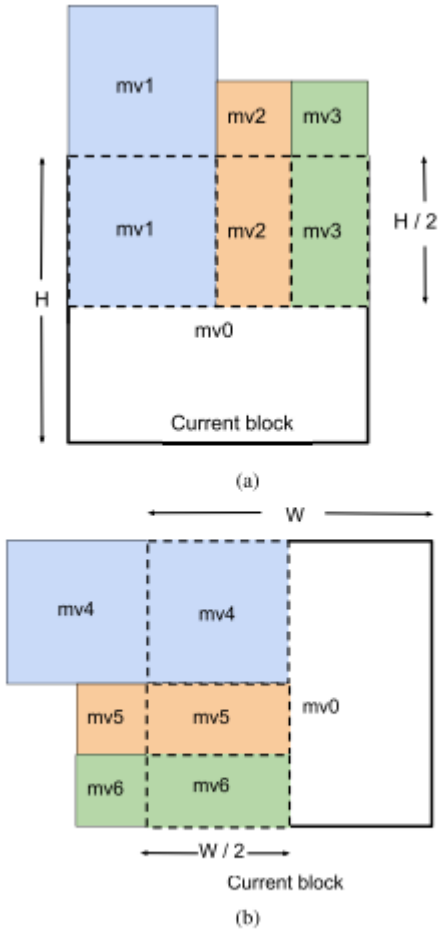


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

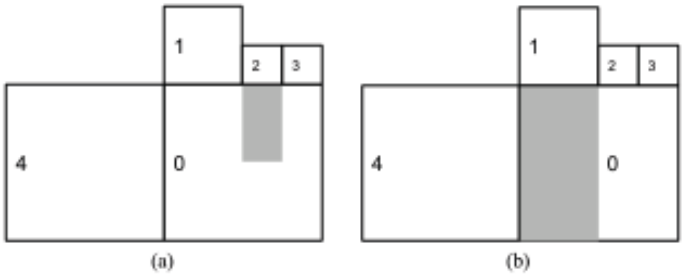


Fig. 1. Overlapping regions

https://ieeexplore.ieee.org/document/8296419

Additionally, for example, the Samsung 8K Neo QLED QN800A, through the use of the AV1 decoder generates border motion-vector data to be coded to the border of the reference picture (*i.e.*, reference frame) because the motion vector is defined as a "two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame." Such a motion vector is generated at each location on an 8x8 luma sample grid. The purpose of such generated motion vectors is to further generate a motion compensation block that extends from the top boundary to toward the center of the current block.

Motion vector

A two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

7.9. Motion field estimation process

7.9.1. General

This process is triggered by a call to motion_field_estimation while reading the uncompressed header.

A linear projection model is employed to create a motion field estimation that is able to capture high velocity temporal motion trajectories.

The motion field is estimated based on the saved motion vectors from the reference frames and the relative frame distances.

As the frame distances depend on the frame being referenced, a separate motion field is estimated for each reference frame used by the current frame.

A motion vector (for each reference frame type) is prepared at each location on an 8x8 luma sample grid.

The variable w8 (representing the width of the motion field in units of 8x8 luma samples) is set equal to MiCols >> 1.

The variable h8 (representing the height of the motion field in units of 8x8 luma samples) is set equal to MiRows >> 1.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 212

e) Overlapped block motion compensation: The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

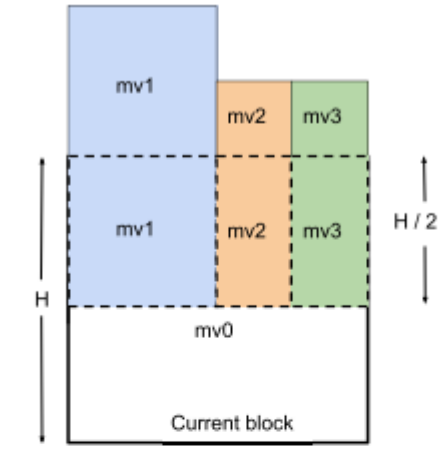


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

https://ieeexplore.ieee.org/document/8296419

	Motion vector coding accounts for a sizable portion of the overall bit rate. Modern video codecs typically adopt predictive coding for motion vectors and code the difference using entropy coding [30], [31]. The prediction accuracy has a large impact on the coding efficiency. AV1 employs a dynamic motion vector referencing scheme that obtains candidate motion vectors from the spatial and temporal neighbors and ranks them for efficient entropy coding. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6A4] defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data, and	The Samsung Accused Products practice: defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec, which is described above with respect to [10A3] and is incorporated here by reference. Specifically, a boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (*i.e.*, immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (*i.e.*, motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6A5] generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing the predictive picture;	The Samsung Accused Products practice: generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing the predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A practices overlapped block motion compensation using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (*i.e.*, prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (*i.e.*, that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

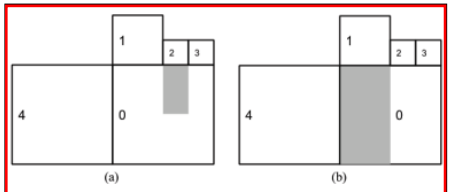


Fig. 1. Overlapping regions

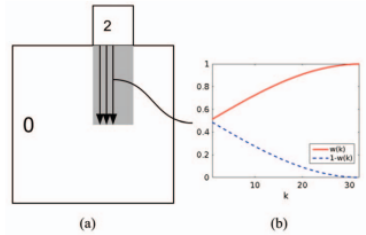


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

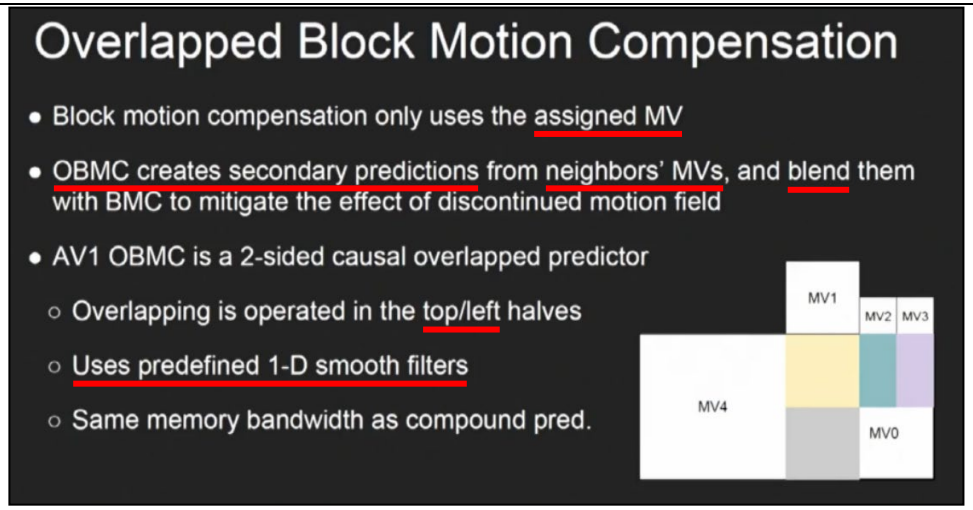
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

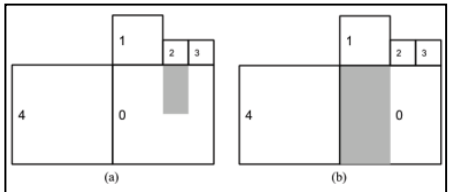


Fig. 1. Overlapping regions

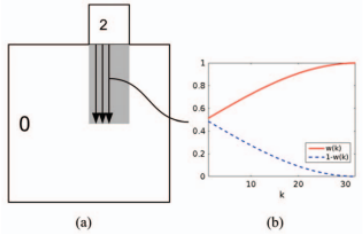


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2-1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

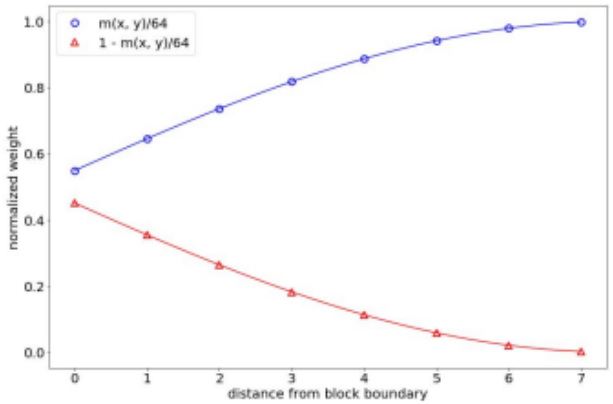


Fig. 15. ***Normalized weights for OBMC with H = 16 or W = 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6B] an entropy decoder to perform entropy decoding to the data thus demultiplexed to generate, at least, motion-vector data, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure;	The Samsung Accused Products comprise an entropy decoder to perform entropy decoding to the data thus demultiplexed to generate, at least, motion-vector data, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A includes an entropy decoder that performs entropy decoding to the demultiplexed data required for constructing a specific syntax structure. Entropy decoding in AV1 generates, at least, the post-quantization data (*i.e.*, quantized transform coefficients), the border motion vector data (*i.e.*, at least the "the motion vector difference [that] will be entropy coded"), and parameter data (*i.e.*, quantization parameters and other "lateral data") for constructing a specific syntax structure (*i.e.*, open bitstream units). Entropy decoding is performed in order to construct a specific syntax structure (*i.e.*, "to compress the syntax elements"). In other words, the "output of this [entropy decoding] process is the variable symbol, containing a decoded syntax element."

employed for transform blocks at dimension 32×32 and above.

F. Quantization

The transform coefficients are quantized, and the quantization indexes are entropy coded. The quantization parameter (QP) in AV1 ranges between 0 and 255.

1) Quantization Step Size: At a given QP, the quantization step size for the dc coefficient is smaller than that for the ac coefficient. The mapping from QP to quantization step size for both dc and ac coefficients is drawn in Fig. 25. The lossless coding mode is achieved when QP is 0. By default, all the ac coefficients will use the same quantization step size. Since the human visual system tends to have different tolerance to distortions at various frequencies, AV1 also supports 15 sets of predefined quantization weighting matrices, where the quantization step size for each individual frequency component is further scaled differently. Each frame can optionally select a quantization weighting matrix set for luma and chroma planes, respectively.

2) Quantization Parameter Modulation: AV1 assigns a base QP for a coded frame, denoted by QP_{base}. The QP values for the dc and ac coefficients in both luma and chroma components are shown in Table 1. $\Delta QP_{p,b}$ are additional offset values transmitted in the frame header,

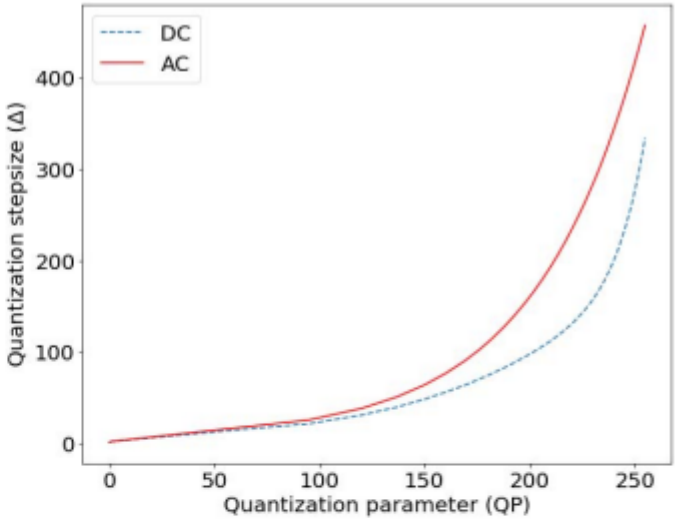


Fig. 25. ***QP and quantization step size maps for dc and ac coefficients.***

where $p \in \{Y, U, V\}$ denotes the plane and $b \in \{dc, ac\}$ denotes the dc or the ac transform coefficients.

Recognizing that the coding blocks within a frame may have different rate-distortion tradeoffs, AV1 further allows QP offset at both superblock and coding block levels. The resolution of QP offset at superblock level is signaled by the frame header. The available options are 1, 2, 4, and 8. The coding block-level QP offset can be achieved through

Table 1 Frame-Level QP Values (QP_{frame}) for Y/U/V Planes

	AC	DC
Y	QP_{base}	$QP_{base} + \Delta QP_{Y,DC}$
U	$QP_{base} + \Delta QP_{U,AC}$	$QP_{base} + \Delta QP_{U,DC}$
V	$QP_{base} + \Delta QP_{V,AC}$	$QP_{base} + \Delta QP_{V,DC}$

segmentation. AV1 allows a frame to classify its coding blocks into up to eight segments; each has its own QP offset decided by the frame header. The segment index associated with each coding block is sent through the bitstream to the decoder.

Therefore, the effective QP for ac coefficients in a coding block, QP_{cb}, is given by

$$QP_{cb} = \text{clip}(QP_{\text{frame}} + \Delta QP_{sb} + \Delta QP_{\text{seg}}, 1, 255) \quad (27)$$

where ΔQP_{sb} and ΔQP_{seg} are the QP offsets from the superblock and the segment, respectively. The clip function ensures it stays within a valid range. The QP is not allowed to change from a nonzero value to zero since zero is reserved for lossless coding.

The decoder rebuilds the quantized samples using a uniform quantizer. Given the quantization step size Δ and the quantization index k, the reconstructed sample is $k\Delta$.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The first four motion vectors in this ranked list will be used as candidate motion vector predictors. The encoder will pick the one that is closest to the desired motion vector and send its index to the decoder. It is not uncommon for coding blocks to have fewer than four candidate motion vectors, due to either the high flexibility in the reference frame selection, or a highly consistent motion activity in the local region. In such a context, the candidate motion vector list will be shorter than 4, which allows the codec to save bits spent on identifying the selected index. The dynamic candidate motion vector list is in contrast to the design in VP9, where one always constructs two candidate motion vectors. If not enough candidates are found, the VP9 codec will fill the list with zero vectors. AV1 also supports a special intermode that makes the inter-predictor use the frame-level affine model, as discussed in Section V-C2.

The motion vector difference will be entropy coded. Since a significant portion of the coding blocks will find a zero motion vector difference, the probability model is designed to account for such bias. AV1 allows a coding block to use 1 bit to indicate whether to directly use the selected motion vector predictor as its final motion vector or to additionally code the difference. The probability model for this entropy-coded bit is conditioned on two factors: whether its spatial neighbors have a nonzero motion vector difference and whether a sufficient number of motion vector predictors are found. For compound modes, where two motion vectors need to be specified, this extends to four cases that cover where either block, both, or neither one have a zero difference motion vector. The nonzero difference motion vector coding is consistent in all cases.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

VI. ENTROPY CODING SYSTEM

AV1 employs an M-ary symbol arithmetic coding method that was originally developed for the Daala video codec [39] to compress the syntax elements, where integer $M \in [2, 14]$. The probability model is updated per symbol coding.

C. Level Map Transform Coefficient Coding System

The transform coefficient entropy coding system is an intricate and performance-critical component in video codecs. We discuss its design in AV1 that decomposes it into a series of symbol codings.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

E. ENTROPY CODING

The entropy coding processes the symbols (quantized coefficients and lateral data) to reduce their statistical redundancy by applying lossless algorithms.

AV1 uses a symbol-to-symbol adaptive multi-symbol arithmetic coder with the probability being updated every new symbol. Each syntax element in AV1 is a member of an alphabet of N elements, and a context consists of a set of N probabilities together with a small count to facilitate fast early adaptation [2].

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

Quantization parameter
A variable used for scaling the quantized coefficients in the decoding process.

Quantized coefficient
A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

D. TRANSFORMS AND QUANTIZATION

The prediction error, or the residues, between the intra and inter prediction and the original blocks are processed by the transform module (T module, in Fig. 1), which converts the values from the spatial domain to the frequency domain. Then, the quantization step (Q module in Fig. 1) is applied to the transformed coefficients to attenuate or eliminate values associated with spectral components that are not perceptually relevant for the human visual system.

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

8.2. Parsing process for symbol decoder

8.2.1. General

The entropy decoder is referred to as the “Symbol decoder” and the functions init_symbol(sz), exit_symbol( ), read_symbol(cdf), and read_bool( ) are used in this Specification to indicate the entropy decoding operation.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

	8.2.2. Initialization process for symbol decoder The input to this process is a variable sz specifying the number of bytes to be read by the Symbol decoder. This process is invoked when the function init_symbol(sz) is called from the syntax structure. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

8.2.4. Exit process for symbol decoder

This process is invoked when the function exit_symbol( ) is called from the syntax structure.

It is a requirement of bitstream conformance that SymbolMaxBits is greater than or equal to -14 whenever this process is invoked.

The variable trailingBitPosition is set equal to get_position() - Min(15, SymbolMaxBits+15).

The bitstream position indicator is advanced by Max(0,SymbolMaxBits). (This skips over any trailing bits that have not already been read during symbol decode.)

The variable paddingEndPosition is set equal to get_position().

> **Note:** paddingEndPosition will always be a multiple of 8 indicating that the bit position is byte aligned.

It is a requirement of bitstream conformance that the bit at position trailingBitPosition is equal to 1.

It is a requirement of bitstream conformance that the bit at position x is equal to 0 for values of x strictly between trailingBitPosition and paddingEndPosition.

> **Note:** This exit process consumes the OBU trailing bits for a tile group.

If disable_frame_end_update_cdf is equal to 0 and TileNum is equal to context_update_tile_id, a copy is made of the final CDF values for each of the CDF arrays mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs. The name of the destination for the copy is the name of the CDF array prefixed with “Saved”. The name of the source for the copy is the name of the CDF array prefixed with “Tile”. For example, an array SavedYModeCdf will be created with values equal to TileYModeCdf.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 358-359

8.2.6. Symbol decoding process

The input to this process is an array cdf of length N + 1 which specifies the cumulative distribution for a symbol with N possible values.

The output of this process is the variable symbol, containing a decoded syntax element. The process also modifies the input array cdf to adapt the probabilities to the content of the stream.

This process is invoked when the function read_symbol(cdf) is called.

> **Note:** When this process is invoked, N will be greater than 1 and cdf[N-1] will be equal to 1 << 15.

The variables cur, prev, and symbol are calculated as follows:

```
cur = SymbolRange
symbol = -1
do {
  symbol++
  prev = cur
  f = ( 1 << 15 ) - cdf[ symbol ]
  cur = ((SymbolRange >> 8) * (f >> EC_PROB_SHIFT)) >> (7 - EC_PROB_SHIFT)
  cur += EC_MIN_PROB * (N - symbol - 1)
} while ( SymbolValue < cur )
```

Note: Implementations may prefer to store the inverse cdf to move the subtraction out of this loop.

The variable SymbolRange is set to prev - cur.

The variable SymbolValue is set equal to SymbolValue - cur.

The range and value are renormalized by the following ordered steps:

1. The variable bits is set to 15 – FloorLog2(SymbolRange). This represents the number of new bits to be added to SymbolValue.
2. The variable SymbolRange is set to SymbolRange << bits.
3. The variable numBits is set equal to Min(bits, Max(0, SymbolMaxBits)). This represents the number of new bits to read from the bitstream.
4. The variable newData is read using the f(numBits) parsing process.
5. The variable paddedData is set equal to newData << (bits - numBits).
6. The variable SymbolValue is set to paddedData ^ (((SymbolValue + 1) << bits) - 1).
7. The variable SymbolMaxBits is set to SymbolMaxBits - bits.

Note: bits may be equal to 0, in which case these ordered steps have no effect.

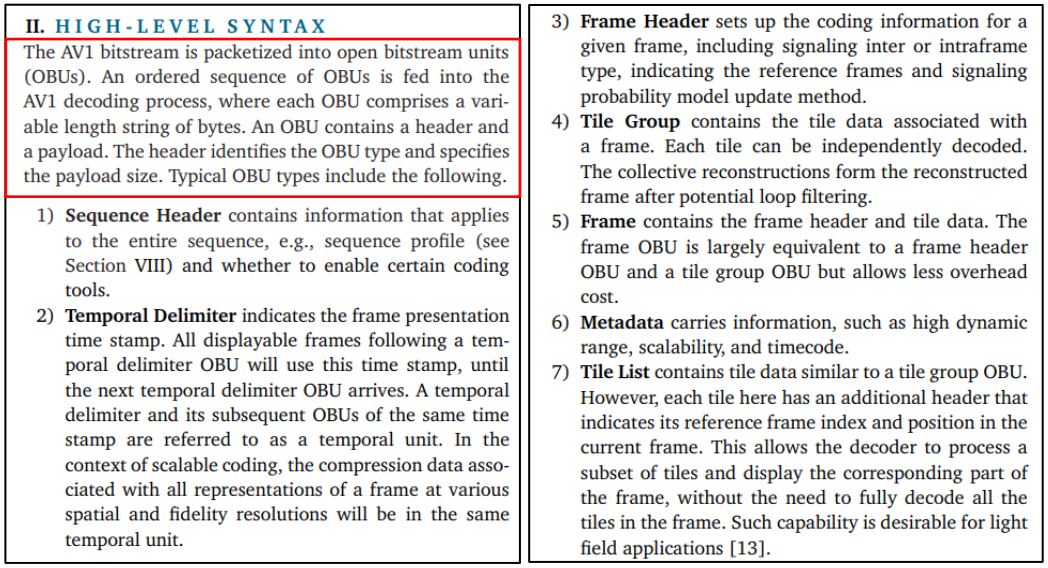
If disable_cdf_update is equal to 0, the cumulative distribution is updated as follows:

```
rate = 3 + ( cdf[ N ] > 15 ) + ( cdf[ N ] > 31 ) + Min( FloorLog2( N ), 2 )
tmp = 0
for ( i = 0; i < N - 1; i++ ) {
  tmp = ( i == symbol ) ? ( 1 << 15 ) : tmp
  if ( tmp < cdf[ i ] ) {
    cdf[ i ] -= ( ( cdf[ i ] - tmp ) >> rate )
  } else {
    cdf[ i ] += ( ( tmp - cdf[ i ] ) >> rate )
  }
}
cdf[ N ] += ( cdf[ N ] < 32 )
```

Note: The last entry of the cdf array is used to keep a count of the number of times the symbol has been decoded (up to a maximum of 32). This allows the cdf adaption rate to depend on the number of times the symbol has been decoded.

The return value from the function is given by symbol.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 359-361



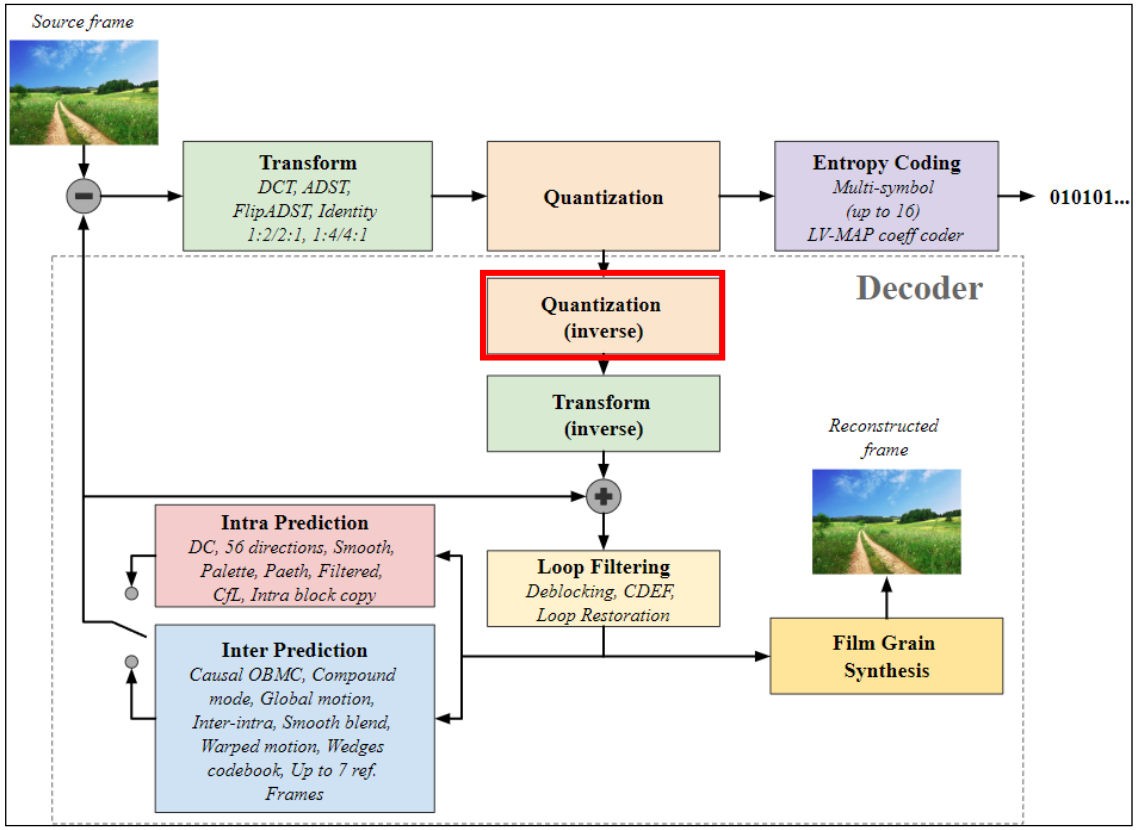
II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6C] an inverse-quantizer to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data;	The Samsung Accused Products comprise an inverse-quantizer to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec as an inverse-quantizer to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Dequantization
The process in which transform coefficients are obtained by scaling the quantized coefficients.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

II. AV1 AND VVC FEATURES

Both the AV1 and VVC encoders follow a hybrid block-based video coding scheme, which is based on the following signal and data processing operations, as presented in Fig. 1: (i) inter and intra-frame prediction, (ii) transform (T), (iii) quantization (Q), and (iv) entropy coding [11]. A reconstruction loop with inverse quantization (IQ) and inverse transform (IT) is also included to guarantee that the encoder and decoder will use the same reference data [11]. Finally,

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12. Reconstruction and dequantization

7.12.1. General

This section details the process of reconstructing a block of coefficients using dequantization and inverse transforms.

7.12.2. Dequantization functions

This section defines the functions get_dc_quant and get_ac_quant that are needed by the dequantization process.

The quantization parameters are derived from lookup tables.

The function get_dc_quant(plane) returns the quantizer value for the dc coefficient for a particular plane and is derived as follows:

- If plane is equal to 0, return dc_q(get_qindex(0, segment_id) + DeltaQYDc).
- Otherwise if plane is equal to 1, return dc_q(get_qindex(0, segment_id) + DeltaQUDc).
- Otherwise (plane is equal to 2), return dc_q(get_qindex(0, segment_id) + DeltaQVDc).

The function get_ac_quant(plane) returns the quantizer value for the ac coefficient for a particular plane and is derived as follows:

- If plane is equal to 0, return ac_q(get_qindex(0, segment_id)).
- Otherwise if plane is equal to 1, return ac_q(get_qindex(0, segment_id) + DeltaQUAc).
- Otherwise (plane is equal to 2), return ac_q(get_qindex(0, segment_id) + DeltaQVAc).

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-294

7.13. Inverse transform process

7.13.1. General

This section details the inverse transforms used during the reconstruction processes detailed in section 7.12.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 296

The generated post-quantization transform coefficients data are orthogonal because at least some of the inverse transform processes are orthogonal (e.g., Inverse Discrete Cosine Transform (DCT) process).

7.13.2.2. Inverse DCT array permutation process

This process performs an in-place permutation of the array T of length 2^n for $2 \le n \le 6$ which is required before execution of the inverse DCT process.

The input to this process is a variable n that specifies the base 2 logarithm of the length of the input array.

A temporary array named copyT is set equal to T.

T[i] is set equal to copyT[brev(n, i)] for i = 0..((1 << n) - 1).

7.13.2.3. Inverse DCT process

This process performs an in-place inverse discrete cosine transform of the permuted array T which is of length 2^n for $2 \le n \le 6$.

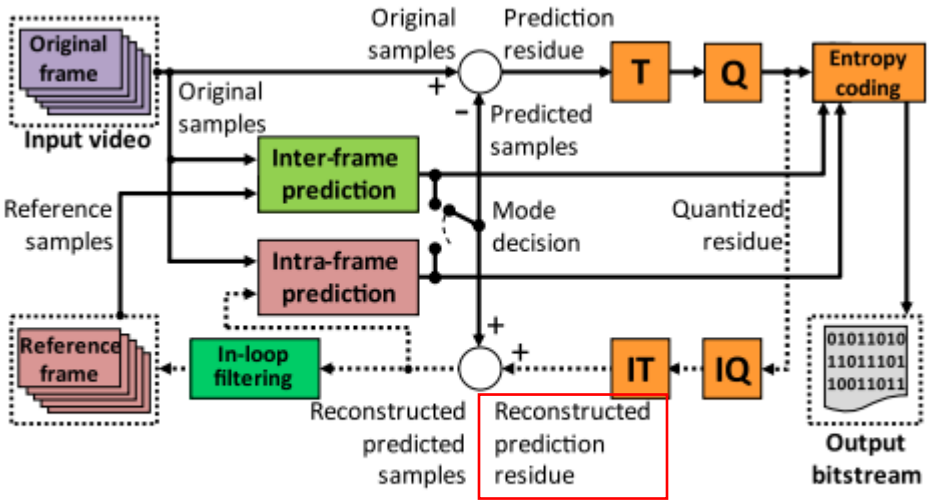
The inputs to this process are:

- a variable n that specifies the base 2 logarithm of the length of the input array,
- a variable r that specifies the intermediate clamping range.

The following ordered steps apply:

1. Invoke the inverse DCT permutation process as specified in section 7.13.2.2 with the input variable n.
2. If n is equal to 6, invoke B(32 + i, 63 - i, 63 - 4 * brev(4, i), 0, r) for i = 0..15.
3. If n is greater than or equal to 5, invoke B(16 + i, 31 - i, 6 + (brev(3, 7 - i) << 3), 0, r) for i = 0..7.
4. If n is equal to 6, invoke H(32 + i * 2, 33 + i * 2, i & 1, r) for i = 0..15.
5. If n is greater than or equal to 4, invoke B(8 + i, 15 - i, 12 + (brev(2, 3 - i) << 4), 0, r) for i = 0..3.
6. If n is greater than or equal to 5, invoke H(16 + 2 * i, 17 + 2 * i, i & 1, r) for i = 0..7.
7. If n is equal to 6, invoke B(62 - i * 4 - j, 33 + i * 4 + j, 60 - 16 * brev(2, i) + 64 * j, 1, r) for i = 0..3, for j = 0..1.
8. If n is greater than or equal to 3, invoke B(4 + i, 7 - i, 56 - 32 * i, 0, r) for i = 0..1.
9. If n is greater than or equal to 4, invoke H(8 + 2 * i, 9 + 2 * i, i & 1, r) for i = 0..3.
10. If n is greater than or equal to 5, invoke B(30 - 4 * i - j, 17 + 4 * i + j, 24 + (j << 6) + ((1 - i) << 5), 1, r) for i = 0..1, for j=0..1.
11. If n is equal to 6, invoke H(32 + i * 4 + j, 35 + i * 4 - j, i & 1, r) for i = 0..7, for j = 0..1.
12. Invoke B(2 * i, 2 * i + 1, 32 + 16 * i, 1 - i, r) for i = 0..1.
13. If n is greater than or equal to 3, invoke H(4 + 2 * i, 5 + 2 * i, i, r) for i = 0..1.
14. If n is greater than or equal to 4, invoke B(14 - i, 9 + i, 48 + 64 * i, 1, r) for i = 0..1.
15. If n is greater than or equal to 5, invoke H(16 + 4 * i + j, 19 + 4 * i - j, i & 1, r) for i = 0..3, for j = 0..1.
16. If n is equal to 6, invoke B(61 - i * 8 - j, 34 + i * 8 + j, 56 - i * 32 + (j >> 1) * 64, 1, r) for i = 0..1, for j = 0..3.
17. Invoke H(i, 3 - i, 0, r) for i = 0..1.
18. If n is greater than or equal to 3, invoke B(6, 5, 32, 1, r).
19. If n is greater than or equal to 4, invoke H(8 + 4 * i + j, 11 + 4 * i - j, i, r) for i = 0..1, for j = 0..1.
20. If n is greater than or equal to 5, invoke B(29 - i, 18 + i, 48 + (i >> 1) * 64, 1, r) for i = 0..3.
21. If n is equal to 6, invoke H(32 + 8 * i + j, 39 + 8 * i - j, i & 1, r) for i = 0..3, for j = 0..3.
22. If n is greater than or equal to 3, invoke H(i, 7 - i, 0, r) for i = 0..3.

	23. If n is greater than or equal to 4, invoke B(13 - i, 10 + i, 32, 1, r) for i = 0..1. 24. If n is greater than or equal to 5, invoke H(16 + i * 8 + j, 23 + i * 8 - j, i, r) for i = 0..1, for j = 0..3. 25. If n is equal to 6, invoke B(59 - i, 36 + i, i < 4 ? 48 : 112, 1, r) for i = 0..7. 26. If n is greater than or equal to 4, invoke H(i, 15 - i, 0, r) for i = 0..7. 27. If n is greater than or equal to 5, invoke B(27 - i, 20 + i, 32, 1, r) for i = 0..3. 28. If n is equal to 6, the following steps apply for i = 0..7: ◦ Invoke H(32 + i, 47 - i, 0, r). ◦ Invoke H(48 + i, 63 - i, 1, r). 29. If n is greater than or equal to 5, invoke H(i, 31 - i, 0, r) for i = 0..15. 30. If n is equal to 6, invoke B(55 - i, 40 + i, 32, 1, r) for i = 0..7. 31. If n is equal to 6, invoke H(i, 63 - i, 0, r) for i = 0..31. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 298-300 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6D] an inverse-orthogonal transformer to perform inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area;	The Samsung Accused Products comprise: an inverse-orthogonal transformer to perform inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec as an inverse-orthogonal transformer to perform an inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data (as described above with respect to [6C], which is incorporated here by reference) to produce a decoded residual picture of one video area (*i.e.*, residual or reconstructed prediction residue).  **FIGURE 1. Block diagram of a typical hybrid video encoder.** https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

The reconstruction and dequantization process is defined as follows:

The variable dqDenom is derived as follows:

- If txSz is equal to TX_32X32, TX_16X32, TX_32X16, TX_16X64, or TX_64X16, dqDenom is set equal to 2.

- Otherwise, if txSz is equal to TX_64X64, TX_32X64, or TX_64X32, dqDenom is set equal to 4.
- Otherwise, dqDenom is set equal to 1.

The variable log2W (specifying the base 2 logarithm of the width of the transform block) is set equal to Tx_Width_Log2[txSz].

The variable log2H (specifying the base 2 logarithm of the height of the transform block) is set equal to Tx_Height_Log2[txSz].

The variable w (specifying the width of the transform block) is set equal to 1 << log2W.

The variable h (specifying the height of the transform block) is set equal to 1 << log2H.

The variable tw is set equal to Min(32, w).

The variable th is set equal to Min(32, h).

The variable flipUD is derived as follows. If PlaneTxType is equal to one of FLIPADST_DCT, FLIPADST_ADST, V_FLIPADST, or FLIPADST_FLIPADST, flipUD is set equal to 1. Otherwise, flipUD is set equal to 0.

The variable flipLR is derived as follows. If PlaneTxType is equal to one of DCT_FLIPADST, ADST_FLIPADST, H_FLIPADST, or FLIPADST_FLIPADST, flipLR is set equal to 1. Otherwise, flipLR is set equal to 0.

The following ordered steps apply:

1. For i = 0..(th-1), for j = 0..(tw-1), the following ordered steps apply:

   a. The variable q is derived as follows:

      - If i is equal to 0 and j is equal to 0, the variable q is set equal to get_dc_quant(plane).
      - Otherwise (i, j or both are not equal to 0), the variable q is set equal to get_ac_quant(plane).

   b. The variable q2 is derived as follows:

      - If using_qmatrix is equal to 1, PlaneTxType is less than IDTX, and SegQMLevel[plane][segment_id] is less than 15, q2 is set equal to Round2(q * Quantizer_Matrix[SegQMLevel[plane][segment_id]][plane > 0][Qm_Offset[txSz] + i * tw + j], 5).
      - Otherwise, q2 is set equal to q.

   c. The variable dq is set equal to Quant[i * tw + j] * q2.

   d. The variable sign is set equal to (dq < 0) ? -1 : 1.

   e. The variable dq2 is set equal to sign * (Abs(dq) & 0xFFFFFF) / dqDenom.

   f. Dequant[i][j] is set equal to Clip3(- (1 << (7 + BitDepth)), (1 << (7 + BitDepth)) - 1, dq2).

2. Invoke the 2D inverse transform block process defined in section 7.13.3 with the variable txSz as input. The inverse transform outputs are stored in the Residual buffer.

3. For i = 0..(h-1), for j = 0..(w-1), the following applies:

    - The variable xx is set equal to flipLR ? (w - j - 1) : j.
    - The variable yy is set equal to flipUD ? (h - i - 1) : i.
    - CurrFrame[plane][y + yy][x + xx] is set equal to Clip1(CurrFrame[plane][y + yy][x + xx] + Residual[i][j]).

If Lossless is equal to 1, it is a requirement of bitstream conformance that the values written into the Residual array in step 2 are representable by a signed integer with 1 + BitDepth bits.

Note: When Lossless is equal to 0, values written into Residual may not be representable by 1 + BitDepth bits (for example, due to quantization noise). The constraints in other parts of the specification ensure that the values will always be representable by a signed integer with Max(BitDepth + 5, 15) bits.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-296

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

	The input to this process is a variable txSz that specifies the transform size. Set the variable log2W equal to Tx_Width_Log2[txSz]. Set the variable log2H equal to Tx_Height_Log2[txSz]. Set the variable w equal to 1 << log2W. Set the variable h equal to 1 << log2H.

Set the variable rowShift equal to Lossless ? 0 : Transform_Row_Shift[txSz].

Set the variable colShift equal to Lossless ? 0 : 4.

Set the variable rowClampRange equal to BitDepth + 8.

Set the variable colClampRange equal to Max(BitDepth + 6, 16).

The row transforms with i = 0..(h-1) are applied as follows:

- T[j] is derived as follows for j = 0..(w-1):
    - If i and j are both less than 32, T[j] is set equal to Dequant[i][j].
    - Otherwise, T[j] is set equal to 0.
- If Abs(log2W - log2H) is equal to 1, T[j] is set equal to Round2(T[j] * 2896, 12) for j = 0..(w-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 2.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, ADST_DCT, FLIPADST_DCT or H_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, if PlaneTxType is equal to one of DCT_ADST, ADST_ADST, DCT_FLIPADST, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, H_ADST, or H_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2W.
- Set Residual[i][j] equal to Round2(T[j], rowShift) for j = 0..(w-1).

Between the row and column transforms, Residual[i][j] is set equal to Clip3(- (1 << (colClampRange - 1)), (1 << (colClampRange - 1)) - 1, Residual[i][j]) for i = 0..(h-1), for j = 0..(w-1).

The column transforms with j = 0..(w-1) are applied as follows:

- Set T[i] equal to Residual[i][j] for i = 0..(h-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 0.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, DCT_ADST, DCT_FLIPADST or V_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2H and the input variable r equal to colClampRange.

- Otherwise, if PlaneTxType is equal to one of ADST_DCT, ADST_ADST, FLIPADST_DCT, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, V_ADST, or V_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2H and the input variable r equal to colClampRange.

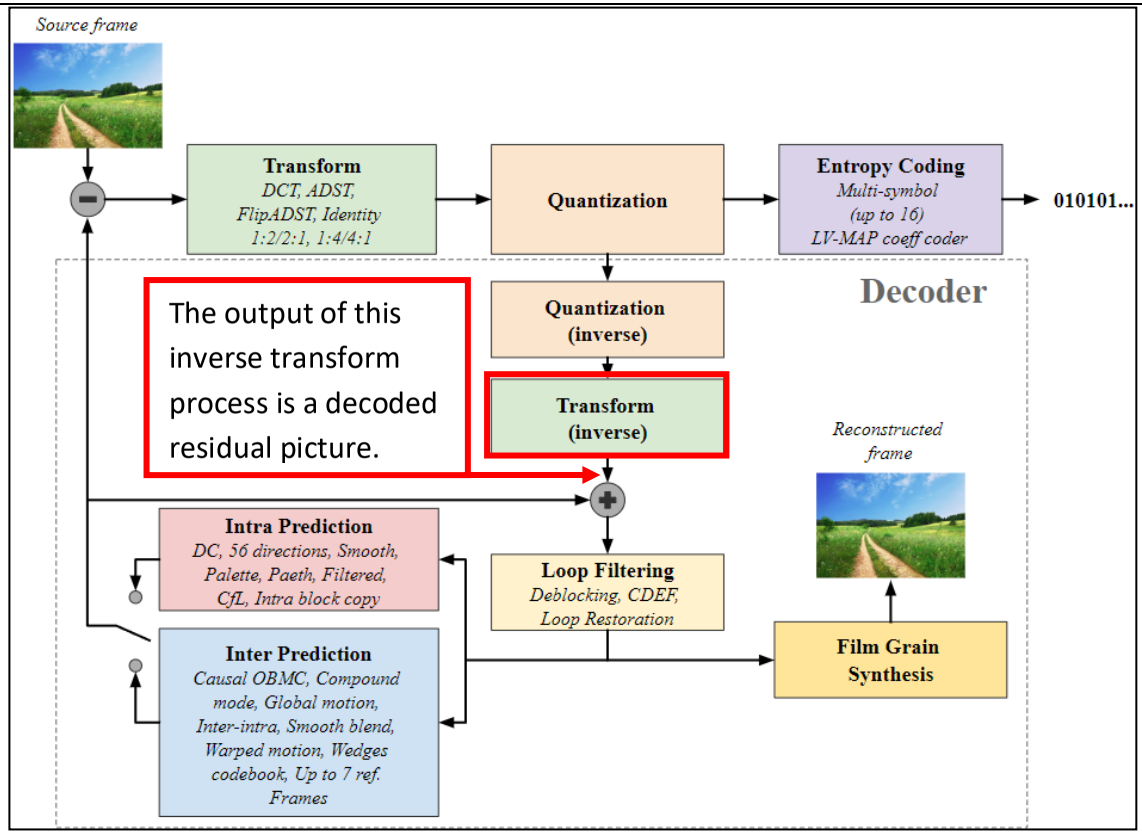
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2H.

- Residual[i][j] is set equal to Round2(T[i], colShift) for i = 0..(h-1).

where Transform_Row_Shift is defined as:

```
Transform_Row_Shift[ TX_SIZES_ALL ] = {
  0, 1, 2, 2, 2, 0, 0, 1, 1,
  1, 1, 1, 1, 1, 1, 2, 2, 2, 2
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305-307



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6E1] a zone-border motion compensator to define a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data, and	The Samsung Accused Products comprise a zone-border motion compensator to define a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* elements [6A3] and [6A4], which are incorporated here by reference. Specifically, the Samsung 8K Neo QLED QN800A defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec. A boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (*i.e.*, immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (*i.e.*, motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6E2] generate an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing a first predictive picture;	The Samsung Accused Products practice: generating an estimated video signal in each rectangular zone in the picture to be decoded, that satisfies Poisson's Equation, thus producing a first predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* element [6A5], which is incorporated here by reference. Specifically, the Samsung 8K Neo QLED QN800A practices overlapped block motion compensation using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (*i.e.*, prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (*i.e.*, that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

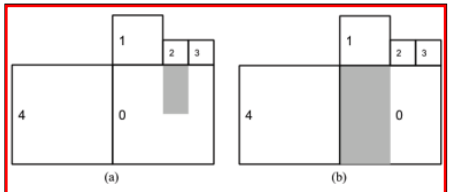


Fig. 1. Overlapping regions

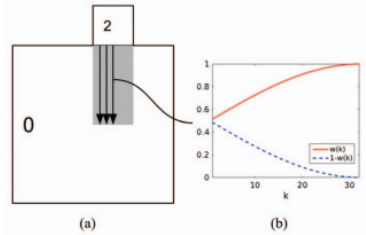


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

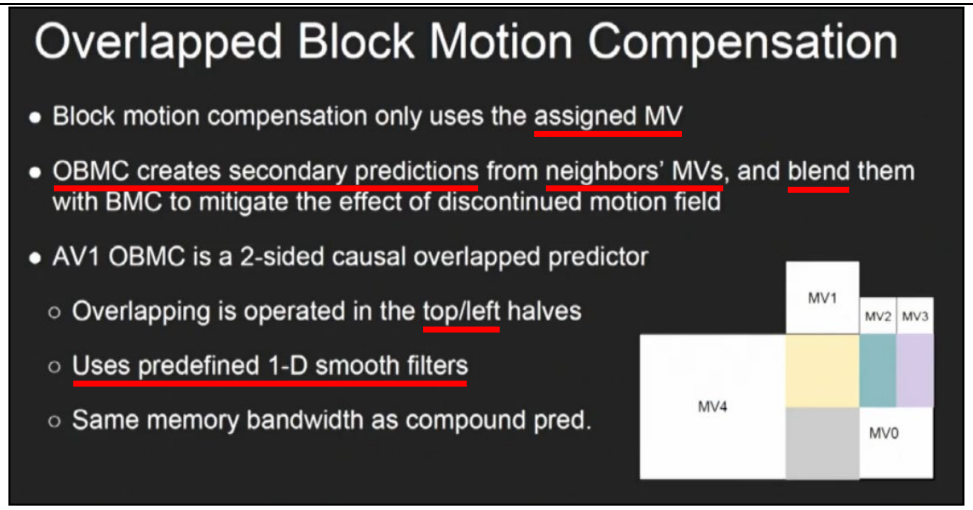
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

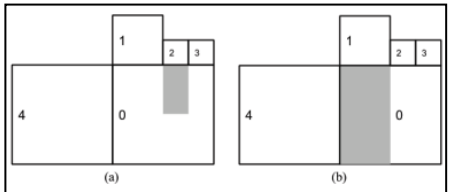


Fig. 1. Overlapping regions

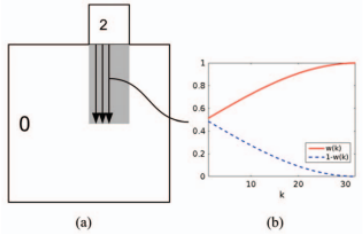


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2-1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

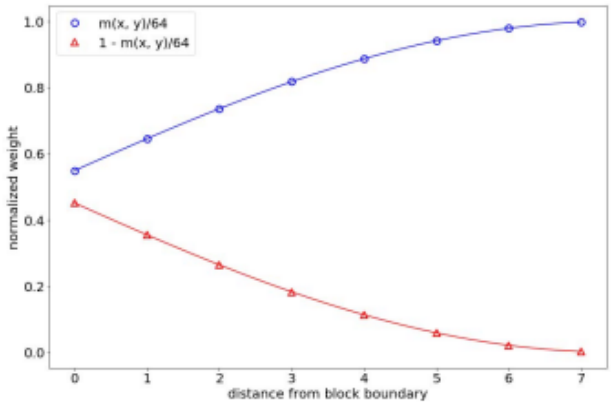


Fig. 15. ***Normalized weights for OBMC with H = 16 or W = 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

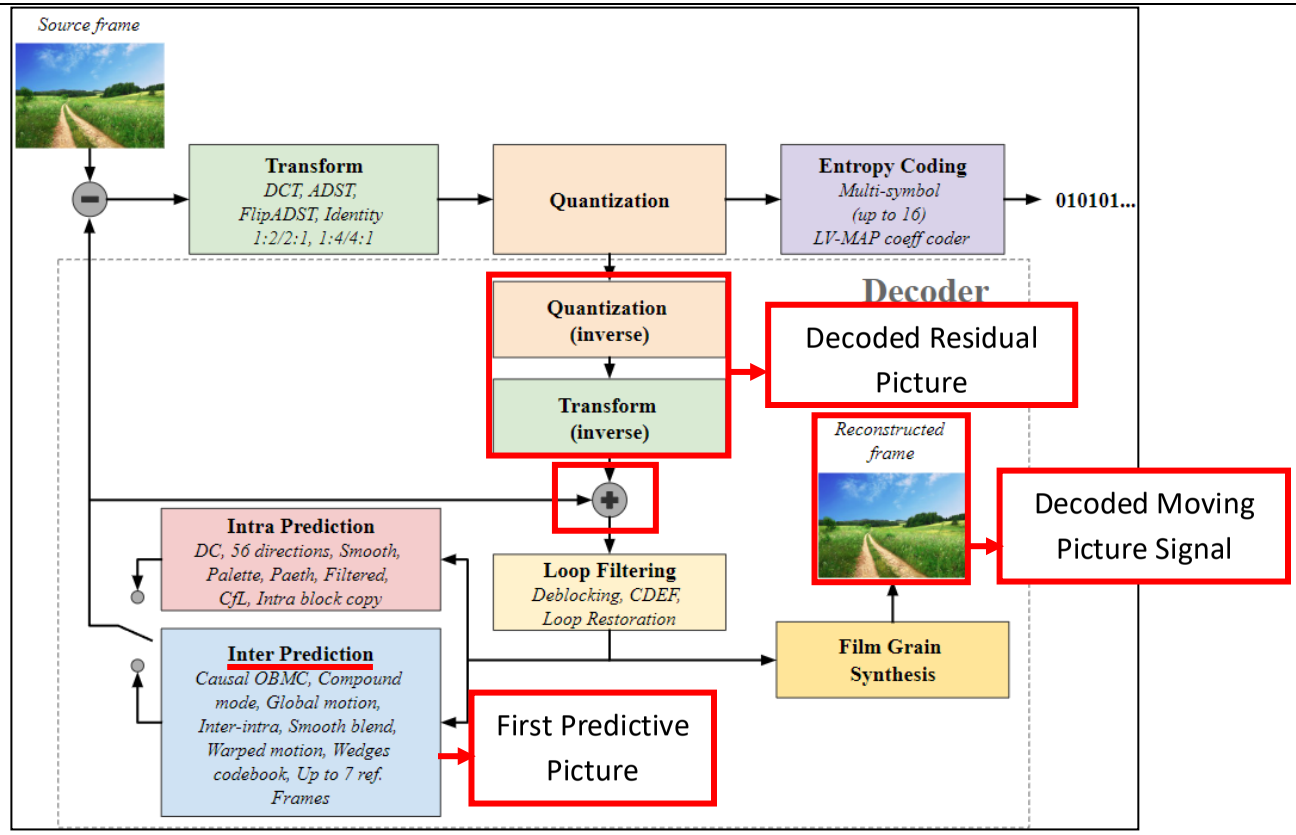
$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6F] a combiner to combine the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal;	The Samsung Accused Products comprise a combiner to combine the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A has a combiner that combines the predictive picture and the decoded residual picture to generate a decoded moving-picture signal using the AV1 codec. Further, to generate a decoded moving picture signal (*i.e.*, the reconstructed frame or reconstructed predicted samples), the AV1 codec combines (*i.e.*, reconstructs) the first predictive picture (*e.g.*, the inter predicted picture or corresponding prediction values) and the decoded residual picture (*i.e.,* the decoded residual, the inverse quantized and inverse transformed picture, or reconstructed prediction residue). **Reconstruction** Obtaining the addition of the decoded residual and the corresponding prediction values. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

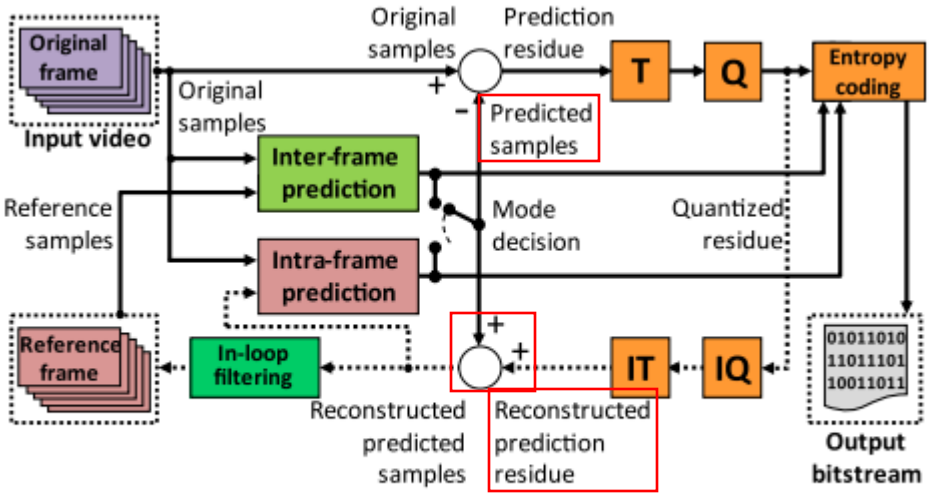


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

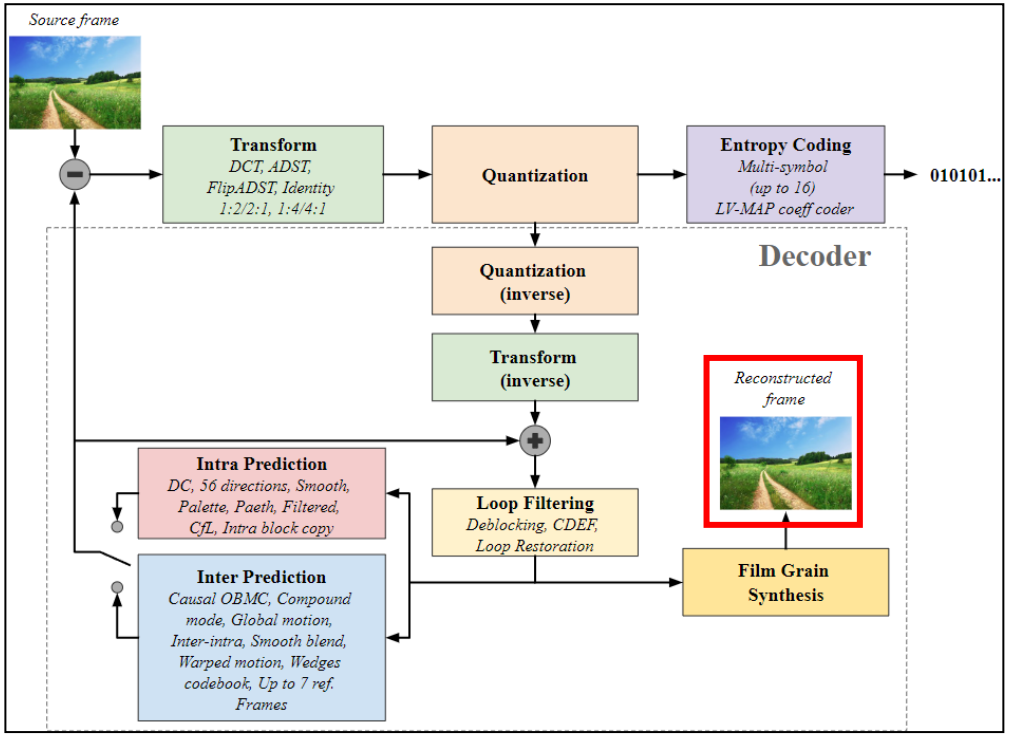
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6G] a memory to store the decoded moving-picture signal for at least one picture as a reference picture;	The Samsung Accused Products comprise: a memory to store the decoded moving-picture signal for at least one picture as a reference picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec to store the decoded moving-picture signal (*i.e.*, decoded samples for the current frame) as reference frames (*i.e.*, an updated set of reference frames) and each of its parameters in memory.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.20. Reference frame update process

This process is invoked as the final step in decoding a frame.

The inputs to this process are the decoded samples for the current frame LrFrame[plane][x][y].

The output from this process is an updated set of reference frames and previous motion vectors.

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefValid[i] is set equal to 1.
- RefFrameId[i] is set equal to current_frame_id.
- RefUpscaledWidth[i] is set equal to UpscaledWidth.
- RefFrameWidth[i] is set equal to FrameWidth.
- RefFrameHeight[i] is set equal to FrameHeight.
- RefRenderWidth[i] is set equal to RenderWidth.
- RefRenderHeight[i] is set equal to RenderHeight.

- RefMiCols[i] is set equal to MiCols.
- RefMiRows[i] is set equal to MiRows.
- RefFrameType[i] is set equal to frame_type.
- RefSubsamplingX[i] is set equal to subsampling_x.
- RefSubsamplingY[i] is set equal to subsampling_y.
- RefBitDepth[i] is set equal to BitDepth.
- SavedOrderHints[i][j + LAST_FRAME] is set equal to OrderHints[j + LAST_FRAME] for j = 0..REFS_PER_FRAME-1.
- FrameStore[i][0][y][x] is set equal to LrFrame[0][y][x] for x = 0..UpscaledWidth-1, for y = 0..FrameHeight-1.
- FrameStore[i][plane][y][x] is set equal to LrFrame[plane][y][x] for plane = 1..2, for x = 0..((UpscaledWidth + subsampling_x) >> subsampling_x) - 1, for y = 0..((FrameHeight + subsampling_y) >> subsampling_y) - 1.
- SavedRefFrames[i][row][col] is set equal to MfRefFrames[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1.
- SavedMvs[i][row][col][comp] is set equal to MfMvs[row][col][comp] for comp = 0..1, for row = 0..MiRows-1, for col = 0..MiCols-1.
- SavedGmParams[i][ref][j] is set equal to gm_params[ref][j] for ref = LAST_FRAME..ALTREF_FRAME, for j = 0..5.
- SavedSegmentIds[i][row][col] is set equal to SegmentIds[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1.
- The function save_cdfs(i) is invoked (see below).
- If film_grain_params_present is equal to 1, the function save_grain_params(i) is invoked (see below).
- The function save_loop_filter_params(i) is invoked (see below).
- The function save_segmentation_params(i) is invoked (see below).

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefOrderHint[i] is set equal to OrderHint.

save_cdfs(ctx) is a function call that indicates that all the CDF arrays are saved into frame context number ctx in the range 0 to (NUM_REF_FRAMES - 1). When this function is invoked the following takes place:

- A copy of each CDF array mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs is saved in an area of memory indexed by ctx.

save_grain_params(i) is a function call that indicates that all the syntax elements that can be read in film_grain_params should be saved into an area of memory indexed by i.

save_loop_filter_params(i) is a function call that indicates that the values of loop_filter_ref_deltas[j] for j = 0 .. TOTAL_REFS_PER_FRAME-1, and the values of loop_filter_mode_deltas[j] for j = 0 .. 1 should be saved into an area of memory indexed by i.

save_segmentation_params(i) is a function call that indicates that the values of FeatureEnabled[j][k] and FeatureData[j][k] for j = 0 .. MAX_SEGMENTS-1, for k = 0 .. SEG_LVL_MAX-1 should be saved into an area of memory indexed by i.

Note: Although this process stores all the motion vectors into SavedMvs, only the values where row and col are both odd will affect the decoding process.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 349-351

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6H] a motion compensator to specify a corresponding rectangular zone in the reference picture based on the motion-vector data, thus generating a second predictive picture;	The Samsung Accused Products comprise: a motion compensator to specify a corresponding rectangular zone in the reference picture based on the motion-vector data, thus generating a second predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a motion compensation program code (*e.g.*, translational motion compensation) to specify a corresponding rectangular zone in the reference picture (*e.g.*, target reference block whose top pixel is located at (x_1, y_1)) based on the motion-vector data (*e.g.*, using the motion vector obtained through motion estimation), thus generating a second predictive picture (*e.g.*, the coding block's motion prediction). *C. Inter Frame Prediction* AV1 supports a rich toolsets to exploit the temporal correlation in video signals. These include adaptive filtering in translational motion compensation, affine motion compensation, and highly flexible compound prediction modes. *1) Translational Motion Compensation:* A coding block uses a motion vector to find its prediction in a reference frame. It first maps its current position, e.g. top-left pixel position (x_0, y_0) in Figure 7, in the reference frame. It is then displaced by the motion vector to the target reference block whose top-left pixel is located at (x_1, y_1).

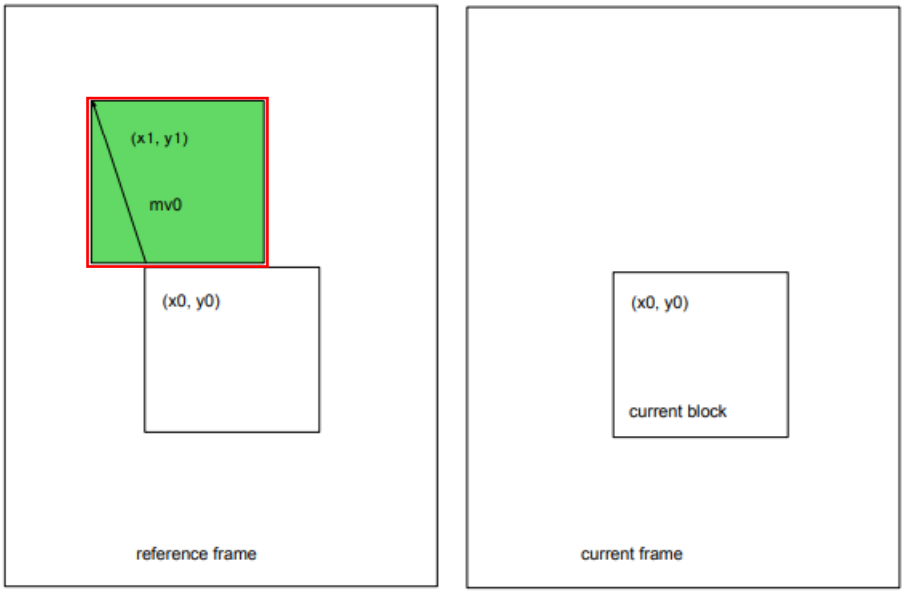


Fig. 7: Translational motion compensated prediction.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6I] a selector to select either the first predictive picture or the second predictive picture and supply the predictive picture thus selected to the combiner; and	The Samsung Accused Products comprise: a selector to select either the first predictive picture or the second predictive picture and supply the predictive picture thus selected to the combiner. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the "motion-mode" flag is a selection program code to select either the first predictive picture (*e.g.*, the predictive picture generated with OBMC) or the second predictive picture (*e.g.*, predictive picture generated with translational motion compensation) and supply the predictive picture thus selected to the combiner. |		
---	---	---
MOTION_MODES	3	Number of values for motion modes
SIMPLE	0	Use translation or global motion compensation
OBMC	1	Use overlapped block motion compensation
LOCALWARP	2	Use local warp motion compensation	 https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 12 6.8.2. Uncompressed header semantics **is_motion_mode_switchable** equal to 0 specifies that only the SIMPLE motion mode will be used. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 154

6.10.26. Read motion mode semantics

use_obmc equal to 1 means that OBMC should be used. use_obmc equal to 0 means that simple translation should be used.

motion_mode specifies the type of motion compensation to perform:

motion_mode	Name of motion_mode
0	SIMPLE
1	OBMC
2	LOCALWARP

Note: A motion_mode equal to SIMPLE is used for blocks requiring global motion.

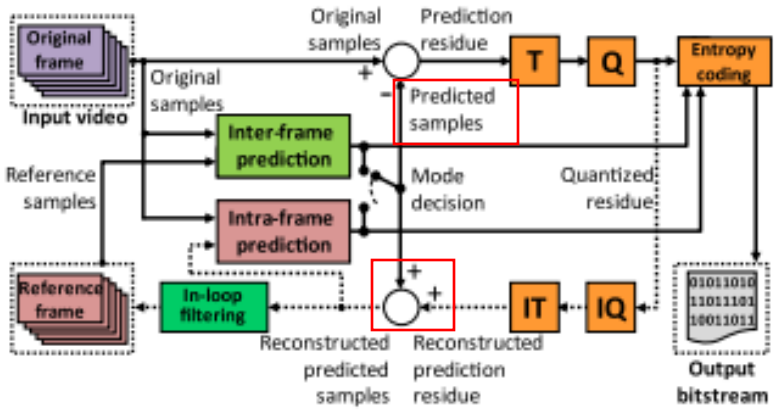
https://aomediacodec.github.io/av1-spec/av1-spec.pdf

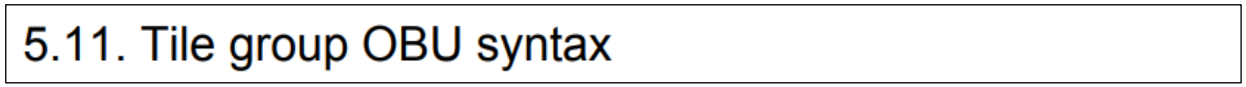
5.11. Tile group OBU syntax

5.11.27. Read motion mode syntax

read_motion_mode(isCompound) {	Type
if (skip_mode) {	
motion_mode = SIMPLE	
return	
}	
if (!is_motion_mode_switchable) {	
motion_mode = SIMPLE	
return	
}	
if (Min(Block_Width[MiSize],	
Block_Height[MiSize]) < 8) {	
motion_mode = SIMPLE	
return	
}	
if (!force_integer_mv &&	
(YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV)) {	
if (GmType[RefFrame[0]] > TRANSLATION) {	
motion_mode = SIMPLE	
return	
}	
}	
if (isCompound || RefFrame[1] == INTRA_FRAME || !has_overlappable_candidates( )) {	
motion_mode = SIMPLE	
return	
}	
find_warp_samples()	
if (force_integer_mv || NumSamples == 0 ||	
!allow_warped_motion || is_scaled(RefFrame[0])) {	
use_obmc	S()
motion_mode = use_obmc ? OBMC : SIMPLE	
} else {	
motion_mode	S()

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	As shown below, the selected predictive picture is provided to the combiner.  FIGURE 1. Block diagram of a typical hybrid video encoder. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8= Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[6J] a decoding controller to receive decoding control data for decoding control from the parameter data and control the selector to switch the predictive picture to be supplied to the combiner between the first and second predictive pictures according to the decoding control data.	The Samsung Accused Products comprise: a decoding controller to receive decoding control data for decoding control from the parameter data and control the selector to switch the predictive picture to be supplied to the combiner between the first and second predictive pictures according to the decoding control data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder uses read motion mode syntax as decoding controlling program code to receive decoding control data for decoding control from the parameter data (*e.g.*, parameters indicating motion mode) and control the selector to switch the predictive picture to be supplied to the combiner between the first (*i.e.*, first predictive picture using OBMC) and second predictive pictures (second predictive picture using translational motion compensation) according to the decoding control data. 

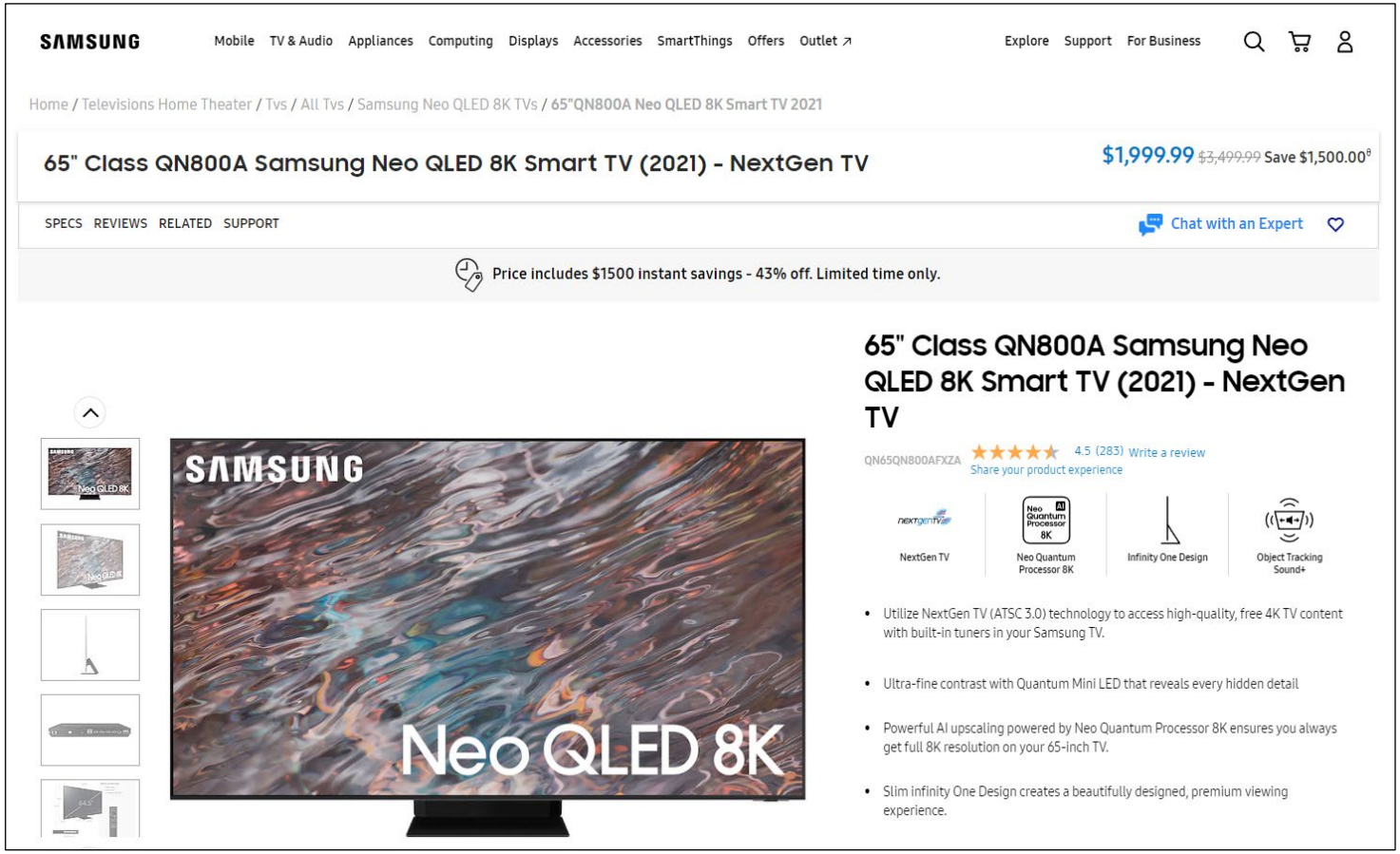
5.11.27. Read motion mode syntax

read_motion_mode(isCompound) {	Type
if (skip_mode) {	
motion_mode = SIMPLE	
return	
}	
if (!is_motion_mode_switchable) {	
motion_mode = SIMPLE	
return	
}	
if (Min(Block_Width[MiSize],	
Block_Height[MiSize]) < 8) {	
motion_mode = SIMPLE	
return	
}	
if (!force_integer_mv &&	
(YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV)) {	
if (GmType[RefFrame[0]] > TRANSLATION) {	
motion_mode = SIMPLE	
return	
}	
}	
if (isCompound || RefFrame[1] == INTRA_FRAME || !has_overlappable_candidates( )) {	
motion_mode = SIMPLE	
return	
}	
find_warp_samples()	
if (force_integer_mv || NumSamples == 0 ||	
!allow_warped_motion || is_scaled(RefFrame[0])) {	
use_obmc	S()
motion_mode = use_obmc ? OBMC : SIMPLE	
} else {	
motion_mode	S()

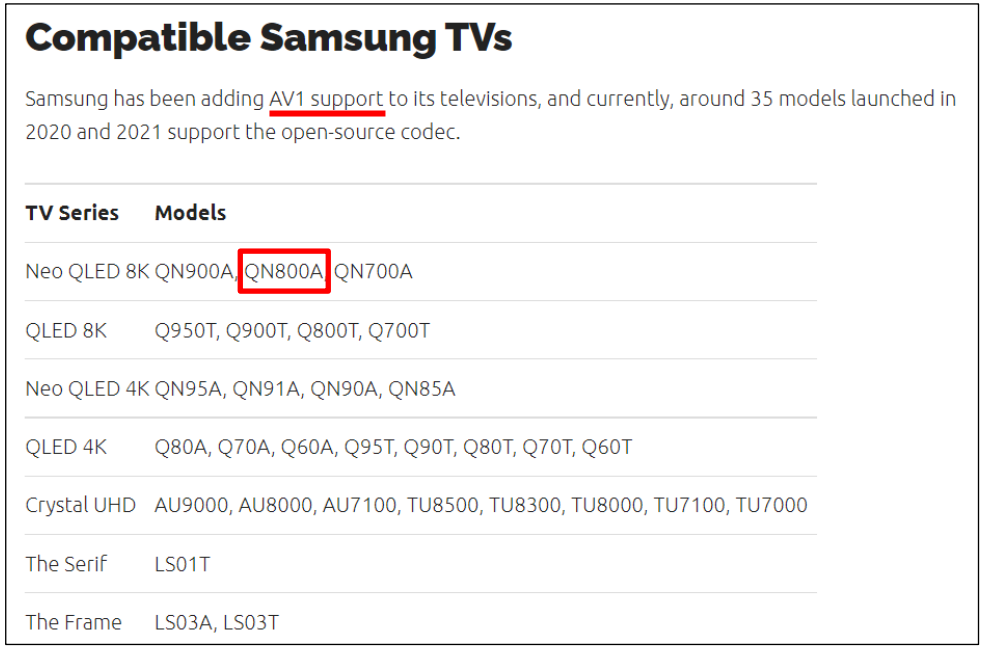
https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7P] A non-transitory computer readable device having stored thereon a computer program comprising a set of instructions when executed by a computer to implement a method for moving-picture decoding, the program comprising:	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising a set of instructions when executed by a computer to implement a method for moving-picture decoding. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A is non-transitory computer readable device having stored thereon a computer program comprising a set of instructions when executed by a computer to implement a method for moving-picture decoding using the AV1 codec.



https://www.samsung.com/us/televisions-home-theater/tvs/samsung-neo-qled-8k/65-qn800a-neo-qled-8k-smart-tv-2021-qn65qn800afxza/



Compatible Samsung TVs

Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec.

TV Series	Models
Neo QLED 8K	QN900A, QN800A, QN700A
QLED 8K	Q950T, Q900T, Q800T, Q700T
Neo QLED 4K	QN95A, QN91A, QN90A, QN85A
QLED 4K	Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T
Crystal UHD	AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000
The Serif	LS01T
The Frame	LS03A, LS03T

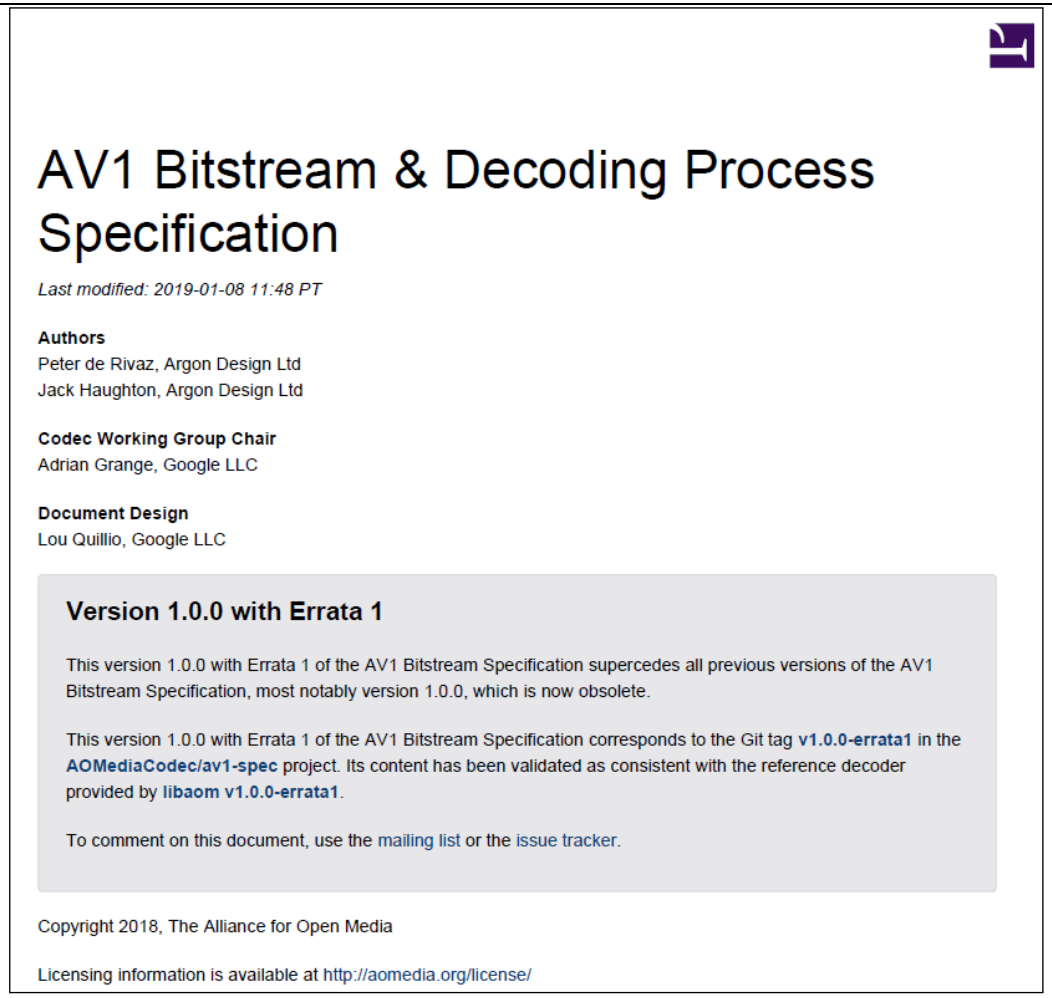
https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

The AV1 codec is used for video decoding, which is moving picture decoding.



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

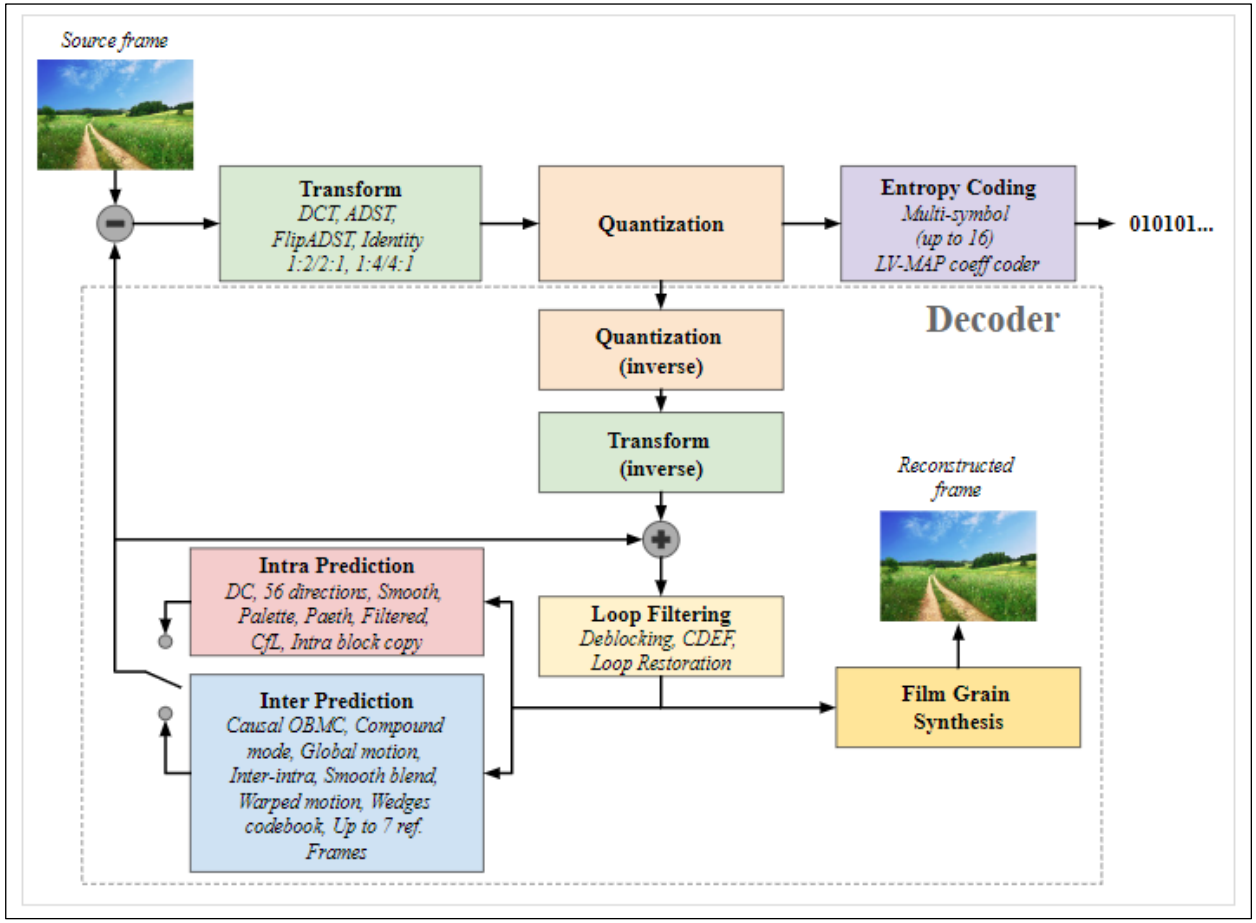
This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	The AV1 decoder is a moving-picture coding apparatus that implements a moving-picture decoding method, as described below to reproduce a reconstructed frame.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Moreover, the AV1 codec comprises a non-transitory computer-readable recording medium for storing its moving picture decoding program. The recording medium is a combination of two forms of memory: DRAM

(dynamic random access memory) and SRAM (static random access memory) with a bus in between the two memories. As an example of these computer-readable memories configured to have a computer execute the AV1 codec's moving picture decoding program (*i.e.*, AV1 video compression), reference frame motion information needs to be transferred to the SRAM from the DRAM "for computing purposes."

***Abstract*—The AV1 video compression format is developed by the Alliance for Open Media consortium. It achieves more than 30% reduction in bit-rate compared to its predecessor VP9 for the same decoded video quality. This paper provides a technical overview of the AV1 codec design that enables the compression performance gains with considerations for hardware feasibility.**

***Index Terms*—AV1, Alliance of Open Media, video compression**

I. Introduction

THE last decade has seen a steady and significant growth of web-based video applications including video-on-demand (VoD) service, live streaming, conferencing, and virtual reality [1]. Bandwidth and storage costs have driven the need for video compression techniques with better compression efficiency. VP9 [2] and HEVC [3], both debuted in 2013, achieved in the range of 50% higher compression performance [4] than the prior codec H.264/AVC [5] and were quickly adopted by the industry.

As the demand for high performance video compression continued to grow, the Alliance for Open Media [6] was formed in 2015 as a consortium for the development of open, royalty-free technology for multimedia delivery. Its first video compression format AV1, released in 2018, enabled about 30% compression gains over its predecessor VP9. The AV1 format is already supported by many web platforms including Android, Chrome, Microsoft Edge, and Firefox and multiple web-based video service providers, including YouTube, Netflix, Vimeo, and Bitmovin, have begun rolling out AV1 streaming services at scale.

	3) Hardware Constraints: The motion information, including the motion vector and the reference frame index, needs to be stored for later frames to build their motion fields. To reduce the memory footprint, the motion information is stored in units of 8×8 blocks. If a coding block is using compound modes, only the first motion vector is saved. The reference frame motion information is commonly stored in the dynamic random access memory (DRAM), a relatively cheaper and slower unit as compared to SRAM, in hardware decoders. It needs, however, to be transferred to SRAM for computing purposes. The bus between DRAM and SRAM is typically 32-bit wide. To facilitate efficient data transfer, a number of data format constraints are employed. We limit the codec to use motion information from up to 4 reference frames (out of 7 available frames) to build the motion field. Therefore only 2 bits are needed for the reference frame index. Furthermore, a motion vector with any component magnitude above 2^{12} will be discarded. As a result, the motion vector and reference frame index together can be represented by a 32-bit unit. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7A1] a demultiplex program code to demultiplex coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding,	The Samsung Accused Products are non-transitory computer readable device having stored thereon a computer program comprising: a demultiplex program code to demultiplex coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises demultiplex program code to demultiplex coded data (.mp4, .mkv, or .webm files) from an input signal based on a specific syntax structure (open bitstream unit (OBU) syntax structure). **What is an AV1 file?** An AV1 file is a video encoded with AV1 compression, which is an open-source video codec developed by the Alliance for Open Media. AV1 videos are more commonly saved with the .MP4, .MKV, and .WEBM extension(s). https://fileinfo.com/extension/av1

2022 TV Video Specifications

Samsung Smart TV provides rich capabilities for video playback, using the available audio and video decoders. This topic describes the media formats supported by Samsung Smart TV, and the limitations on their support.

22TV_PREMIUM1 (8K)

File Extension	Container	Video Codec	Resolution	Frame Rate (fps)	Bit Rate (Mbps)	Audio Codec
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS	H.264 BP/MP/HP	4096x2160	60	60	AC3 LPCM ADPCM(IMA, MS) AAC HE-AAC WMA DD+ MPEG(MP3) MPEG-H AC-4 OPUS G.711(A-Law, μ-Law)
			1920x1080	120		
		HEVC (H.265 - Main, Main10)	7680x4320	60	100	
			3840x2160	120		
		Motion JPEG		30	80	
		MPEG-4 SP/ASP	1920x1080	60	20	
		Window Media Video 9 (VC-1)				
		MPEG-2				
		MPEG-1				
		Microsoft MPEG-4 version 1/2/3		30		
		Window Media Video 7 (WMV1) / Video 8 (WMV2)				
		H.263 Sorenson				
		VP6				
		AV1	7680x4320	60	80	
				120		
*.webm	WebM	VP8	1920x1080	60	20	Vorbis
		VP9 (Profile 0/2)	3840x2160		80	
		AV1	7680x4320		60	
*.rmvb	RMVB	RV8/9/10 (RV30/40)	1920x1080		20	RealAudio 6

https://developer.samsung.com/smarttv/develop/specifications/media-specifications/2022-tv-video-specifications.html

II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

We refer to [9] for bit field definitions and more detailed consideration of high-level syntax.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 input signal to Samsung TVs is obtained by multiplexing a coded bitstream obtained by predictive coding (*i.e.*, predictive-transform coding), border motion-vector data (*i.e.*, interframe motion compensation), and post-quantization data (*i.e.*, reconstructed samples based on dequantized coefficients and inverse transforms)) obtained by quantization in the predictive coding.

The AV1 codec supports input video signals in the 4:0:0 (monochrome), 4:2:0, 4:2:2, and 4:4:4 formats. The allowed pixel representations are 8, 10, and 12 bit. The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D unitary transform to further remove the spatial correlations, and the transform coefficients are quantized. Both the prediction syntax elements and the quantized transform coefficient indexes are then entropy coded using arithmetic coding. There are three optional in-loop postprocessing filter stages to enhance the quality of the reconstructed frame for reference by subsequent coded frames. A normative film grain synthesis unit is also available to improve the perceptual quality of the displayed frames.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

As shown below, the AV1 input signal is obtained by predictive coding (*i.e.*, predictive-transform coding). One example of predictive coding is through inter frame prediction:

V. CODING BLOCK OPERATIONS

C. Inter Frame Prediction

Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector.

Overlapped Block Motion Compensation The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

The variable isCompound is set equal to RefFrames[candRow][candCol][1] > INTRA_FRAME.

The prediction arrays are formed by the following ordered steps:

1. The rounding variables derivation process specified in section 7.11.3.2 is invoked with the variable isCompound as input.
2. If plane is equal to 0 and motion_mode is equal to LOCALWARP, the warp estimation process in section 7.11.3.8 is invoked.
3. If plane is equal to 0 and motion_mode is equal to LOCALWARP and LocalValid is equal to 1, the setup shear process specified in section 7.11.3.6 is invoked with LocalWarpParams as input, and the output warpValid is assigned to LocalValid (the other outputs are discarded).
4. The variable refList is set equal to 0.
5. The variable refFrame is set equal to RefFrames[candRow][candCol][refList].

6. If (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV) and GmType[refFrame] > TRANSLATION, the setup shear process specified in section 7.11.3.6 is invoked with gm_params[refFrame] as input, and the output warpValid is assigned to globalValid (the other outputs are discarded).

7. The variable useWarp (a value of 1 indicates local warping, 2 indicates global warping) is derived as follows:

    - If w < 8 or h < 8, useWarp is set equal to 0.
    - Otherwise, if force_integer_mv is equal to 1, useWarp is set equal to 0.
    - Otherwise, if motion_mode is equal to LOCALWARP and LocalValid is equal to 1, useWarp is set equal to 1.
    - Otherwise, if all of the following are true, useWarp is set equal to 2.
        - (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV).
        - GmType[refFrame] > TRANSLATION.
        - is_scaled(refFrame) is equal to 0.
        - globalValid is equal to 1.
    - Otherwise, useWarp is set equal to 0.

8. The motion vector array mv is set equal to Mvs[candRow][candCol][refList].

9. The variable refIdx specifying which reference frame is being used is set as follows:

    - If use_intrabc is equal to 0, refIdx is set equal to ref_frame_idx[refFrame - LAST_FRAME].
    - Otherwise (use_intrabc is equal to 1), refIdx is set equal to -1 and RefFrameWidth[-1] is set equal to FrameWidth, RefFrameHeight[-1] is set equal to FrameHeight, and RefUpscaledWidth[-1] is set equal to UpscaledWidth. (These values ensure that the motion vector scaling has no effect.)

10. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, x, y, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY.

11. If use_intrabc is equal to 1, RefFrameWidth[-1] is set equal to MiCols * MI_SIZE, RefFrameHeight[-1] is set equal to MiRows * MI_SIZE, and RefUpscaledWidth[-1] is set equal to MiCols * MI_SIZE. (These values are needed to avoid intrabc prediction being cropped to the frame boundaries.)

12. If useWarp is not equal to 0, the block warp process in section 7.11.3.5 is invoked with useWarp, plane, refList, x, y, i8, j8, w, h as inputs and the output is merged into the 2D array preds[refList] for i8 = 0..((h-1) >> 3) and for j8 = 0..((w-1) >> 3). (Each invocation fills in a block of output of size w by h at x offset j8 * 8 and y offset i8 * 8.)

13. If useWarp is equal to 0, the block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, w, h, candRow, candCol as inputs and the output is assigned to the 2D array preds[refList].

14. If isCompound is equal to 1, then the variable refList is set equal to 1 and steps 5 to 13 are repeated to form the prediction for the second reference.

An array named Mask is prepared as follows:

- If compound_type is equal to COMPOUND_WEDGE and plane is equal to 0, the wedge mask process in section 7.11.3.11 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_INTRA, the intra mode variant mask process in section 7.11.3.13 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_DIFFWTD and plane is equal to 0, the difference weight mask process in section 7.11.3.12 is invoked with preds, w, h as inputs.
- Otherwise, no mask array is needed.

If compound_type is equal to COMPOUND_DISTANCE, the distance weights process in section 7.11.3.15 is invoked with candRow and candCol as inputs.

The inter predicted samples are then derived as follows:

- If isCompound is equal to 0 and IsInterIntra is equal to 0, CurrFrame[plane][y + i][x + j] is set equal to Clip1(preds[0][i][j]) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_AVERAGE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(preds[0][i][j] + preds[1][i][j], 1 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_DISTANCE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(FwdWeight * preds[0][i][j] + BckWeight * preds[1][i][j], 4 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise, the mask blend process in section 7.11.3.14 is invoked with preds, plane, x, y, w, h as inputs.

If motion_mode is equal to OBMC, the overlapped motion compensation in section 7.11.3.9 is invoked with plane, w, h as inputs.

7.11.3.4. Block inter prediction process

The inputs to this process are:

- a variable plane,
- a variable refIdx specifying which reference frame is being used (or -1 for intra block copy),
- variables x and y giving the block location with in units of 1/1024 th of a sample,
- variables xStep and yStep giving the step size in units of 1/1024 th of a sample,
- variables w and h giving the width and height of the block in units of samples,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The output from this process is the 2D array named pred containing inter predicted samples.

A variable ref specifying the reference frame contents is set as follows:

- If refIdx is equal to -1, ref is set equal to CurrFrame.
- Otherwise (refIdx is greater than or equal to 0), ref is set equal to FrameStore[refIdx].

The variables subX and subY are set equal to the subsampling for the current plane as follows:

- If plane is equal to 0, subX is set equal to 0 and subY is set equal to 0.
- Otherwise, subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The variable lastX is set equal to ((RefUpscaledWidth[refIdx] + subX) >> subX) - 1.

The variable lastY is set equal to ((RefFrameHeight[refIdx] + subY) >> subY) - 1.

(lastX and lastY specify the coordinates of the bottom right sample of the reference plane.)

The variable intermediateHeight specifying the height required for the intermediate array is set equal to (((h - 1) * yStep + (1 << SCALE_SUBPEL_BITS) - 1) >> SCALE_SUBPEL_BITS) + 8.

The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. The fractional parts of the motion vectors determine the filtering process. If the fractional part is zero, then the filtering is equivalent to a straight sample copy.

The filtering is applied as follows:

- The array intermediate is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 1 ]
if ( w <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < intermediateHeight; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = x + xStep * c
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              ref[ plane ] [ Clip3( 0, lastY, (y >> 10) + r - 3 ) ]
                           [ Clip3( 0, lastX, (p >> 10) + t - 3 ) ]
        intermediate[ r ][ c ] = Round2(s, InterRound0)
    }
}
```

- The array pred is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 0 ]
if ( h <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < h; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = (y & 1023) + yStep * r
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              intermediate[ (p >> 10) + t ][ c ]
        pred[ r ][ c ] = Round2(s, InterRound1)
    }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257-263

As shown below, the AV1 input signal is obtained by border motion-vector data (*i.e.*, interframe motion compensation):

V. CODING BLOCK OPERATIONS

C. Inter Frame Prediction

Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector.

Overlapped Block Motion Compensation The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

	When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector: 1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs.

The function get_obmc_mask returns a blending mask as follows:

```
get_obmc_mask( length ) {
    if ( length == 2 ) {
        return Obmc_Mask_2
    } else if ( length == 4 ) {
        return Obmc_Mask_4
    } else if ( length == 8 ) {
        return Obmc_Mask_8
    } else if ( length == 16 ) {
        return Obmc_Mask_16
    } else {
        return Obmc_Mask_32
    }
}
```

The blending masks are defined as follows:

```
Obmc_Mask_2[2] = { 45, 64 }

Obmc_Mask_4[4] = { 39, 50, 59, 64 }

Obmc_Mask_8[8] = { 36, 42, 48, 53, 57, 61, 64, 64 }

Obmc_Mask_16[16] = { 34, 37, 40, 43, 46, 49, 52, 54,
                     56, 58, 60, 61, 64, 64, 64, 64 }

Obmc_Mask_32[32] = { 33, 35, 36, 38, 40, 41, 43, 44,
                     45, 47, 48, 50, 51, 52, 53, 55,
                     56, 57, 58, 59, 60, 60, 61, 62,
                     64, 64, 64, 64, 64, 64, 64, 64 }
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-278

As shown below, the AV1 input signal is obtained by post-quantization data (i.e., reconstructed samples based on dequantized coefficients and inverse transforms) obtained by quantization (i.e., using transform coefficients to discretize the signal) in the predictive coding:

Transform coefficient

A scalar value, considered to be in a frequency domain, contained in a transform block.

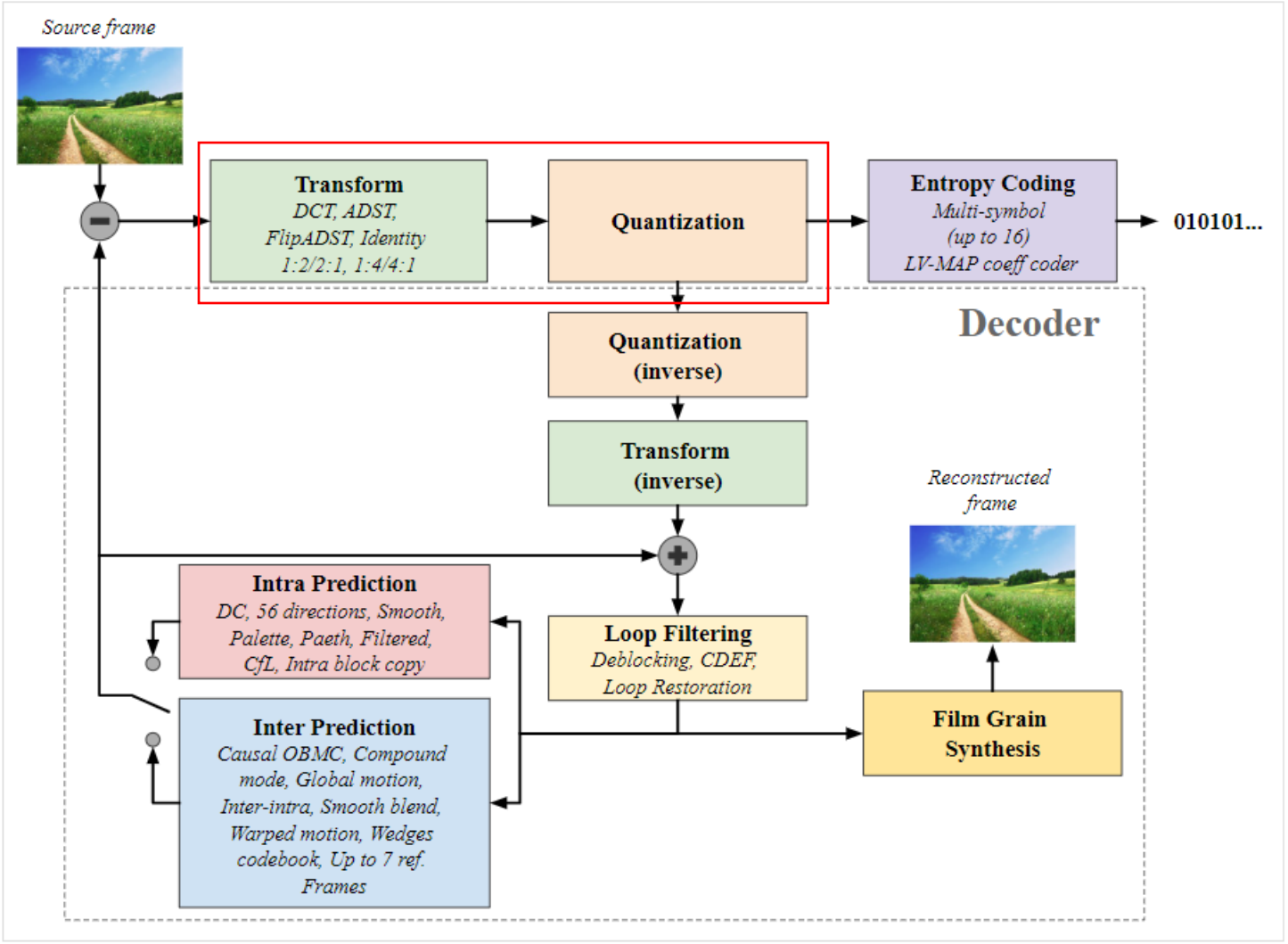
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 6

Quantized coefficient

A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4

	Dequantization The process in which transform coefficients are obtained by scaling the quantized coefficients. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7A2] the coded bitstream obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video	The coded bitstream is obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the coded bitstream input to the Samsung 8K Neo QLED QN800A is obtained by producing and encoding a residual picture that is a residual signal (*i.e.*, prediction residue) between a picture to be coded (*i.e.*, original samples) that is an input moving-picture video signal to be subjected to coding and a predictive picture (*i.e.*, predicted samples) produced from a reference picture (*i.e.*, reference frame).

area of the moving-picture video signal is divided,

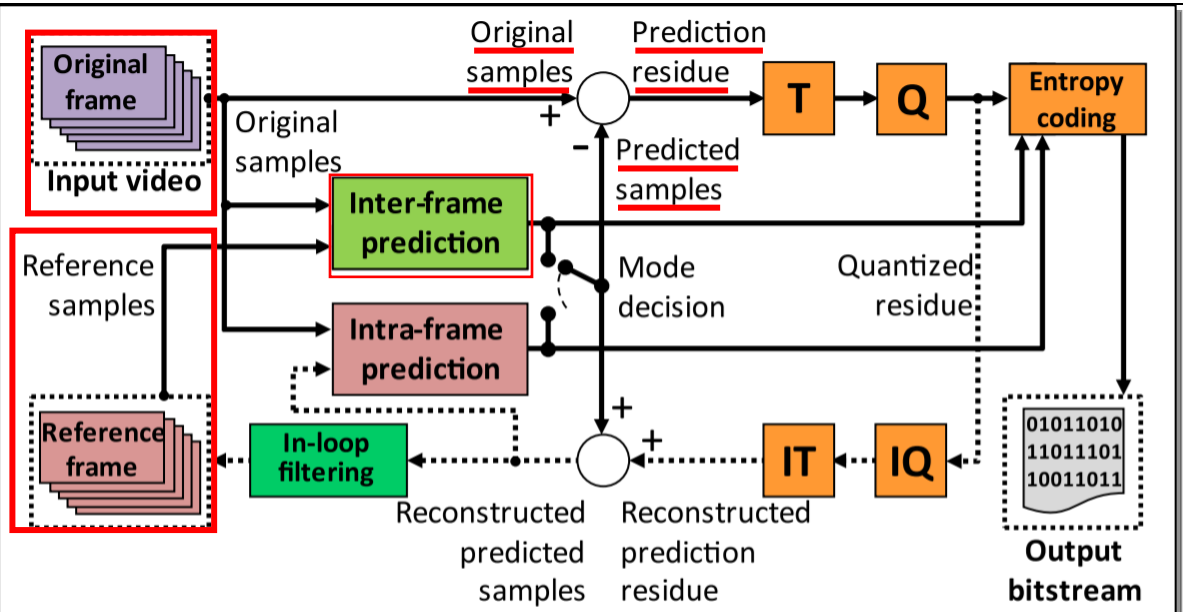


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

The reference picture (*i.e.*, reference frame) is a local decoded video signal (*i.e.*, previously decoded frame) for each of a plurality of rectangular zones (*i.e.*, block). The video area of the moving-picture video signal (*i.e.*, frame) is divided into such rectangular zones (*i.e.*, blocks).

Reference frame
A storage area for a previously decoded frame and associated information.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5

	Frame The representation of video signals in the spatial domain, composed of one luma sample matrix (Y) and two chroma sample matrices (U and V). https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2 **Block** A square or rectangular region of samples. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 1 **Sample** The basic elements that compose the frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5 **Inter coding** Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 3 The plurality of rectangular zones (*i.e.*, several blocks of pixels) are composed of a specific number of pixels (*e.g.*, 128 x 128 pixels, 64 x 64 pixels, 4 x 4 pixels).

V. CODING BLOCK OPERATIONS

A. Coding Block Partitioning

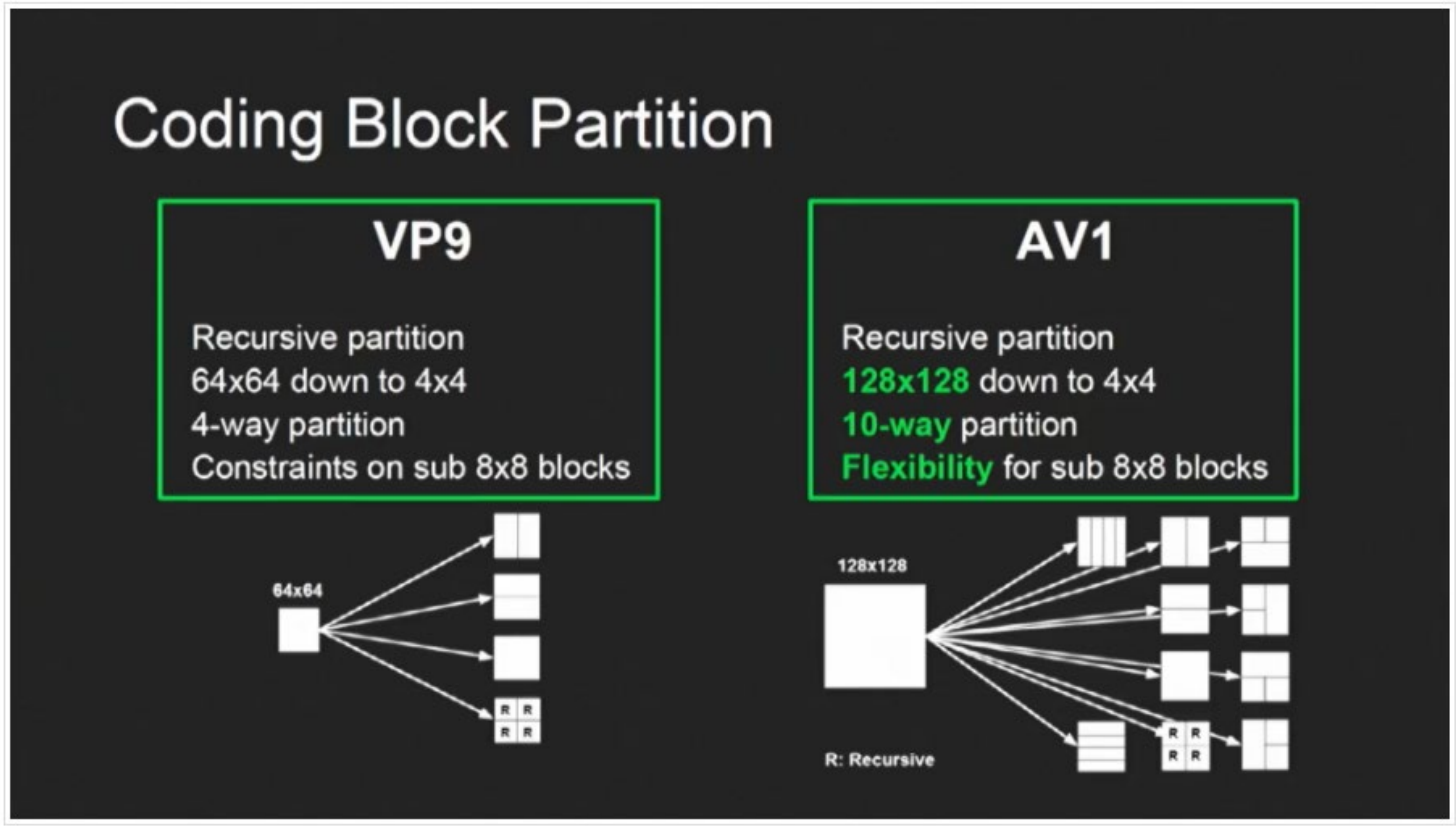
A superblock can be recursively partitioned into smaller block sizes for coding. AV1 inherits the recursive block partitioning design used in VP9. To reduce the overhead cost on prediction mode coding for video signals that are highly correlated, a situation typically seen in 4k videos, AV1 supports a maximum coding block size of 128×128 luma samples. The allowed partition options at each block level include ten possibilities, as shown in Fig. 2. To improve the prediction quality for complex videos, the minimum coding block size is extended to 4×4 luma samples. While such extensions provide more coding flexibility, they have implications for hardware decoders. Certain block size-dependent constraints are specifically designed to circumvent such complications.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D uni-

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	A. BLOCK PARTITIONING Before the prediction process begins, a frame is divided into several blocks of pixels. Each video encoder defines a variable range of block sizes it can use. In AV1, a video frame is partitioned in superblocks (SBs) of size 128×128 or 64×64 pixels. To deliver a locally optimal prediction for each SB, the encoder can further divide each SB using a 10-way partition tree structure, as illustrates Fig. 2. In the figure, filled partitions are final partition modes, but all four partitions of the unfilled partition (SPLIT) can be recursively divided based on the same 10-way tree structure, down to 4×4, which is the smallest supported block size [2]. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

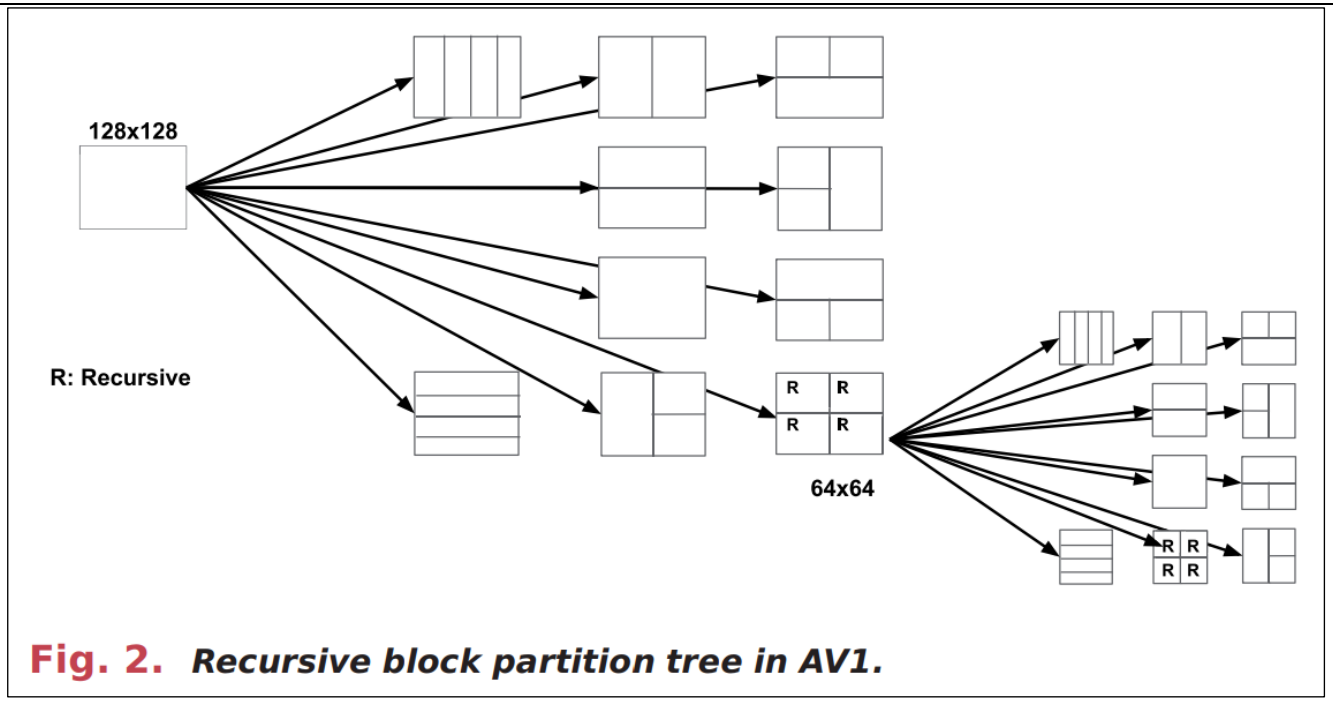


Fig. 2. ***Recursive block partition tree in AV1.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

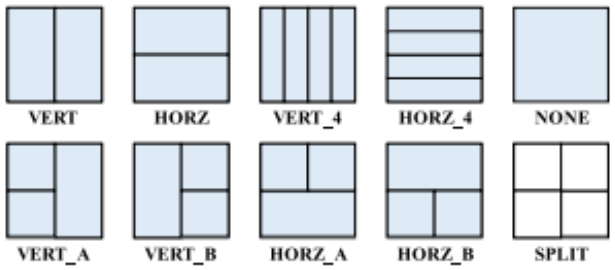


FIGURE 2. AV1 10-way block partition tree structure.

	https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8= Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7A3] obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program which obtains a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A, through the use of the AV1's block-matching algorithms, such as overlapped block motion compensation (OBMC), obtains the boundary conditions of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones (*i.e.*, "immediate spatial neighbors' motion information to improve the prediction quality for pixels near [a block's] top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides"). After the boundary condition of the block is obtained, the OBMC finds a boundary condition in the reference frame that matches the boundary condition in the current frame to identify the "four reference blocks that have the same reference frame as the current block" and ensure that the boundary condition is met (*i.e.*, "width is the same as the reference block's width, and its height is half of the current block's height").

is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found,	*e) Overlapped block motion compensation:* The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as $$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \quad (15)$$ where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function: $$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \quad (16)$$ where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

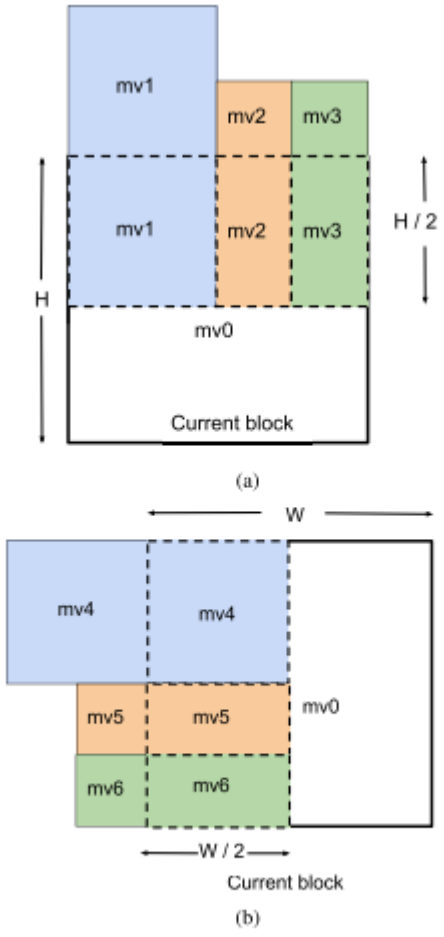


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

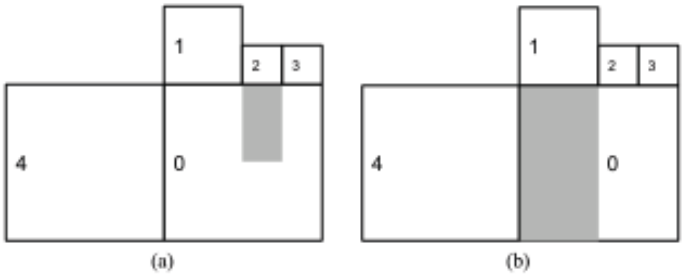


Fig. 1. Overlapping regions

https://ieeexplore.ieee.org/document/8296419

Additionally, for example, the Samsung 8K Neo QLED QN800A, through the use of the AV1 decoder generates border motion-vector data to be coded to the border of the reference picture (*i.e.*, reference frame) because the motion vector is defined as a "two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame." Such a motion vector is generated at each location on an 8x8 luma sample grid. The purpose of such generated motion vectors is to further generate a motion compensation block that extends from the top boundary to toward the center of the current block.

Motion vector

A two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

7.9. Motion field estimation process

7.9.1. General

This process is triggered by a call to motion_field_estimation while reading the uncompressed header.

A linear projection model is employed to create a motion field estimation that is able to capture high velocity temporal motion trajectories.

The motion field is estimated based on the saved motion vectors from the reference frames and the relative frame distances.

As the frame distances depend on the frame being referenced, a separate motion field is estimated for each reference frame used by the current frame.

A motion vector (for each reference frame type) is prepared at each location on an 8x8 luma sample grid.

The variable w8 (representing the width of the motion field in units of 8x8 luma samples) is set equal to MiCols >> 1.

The variable h8 (representing the height of the motion field in units of 8x8 luma samples) is set equal to MiRows >> 1.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 212

e) Overlapped block motion compensation: The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

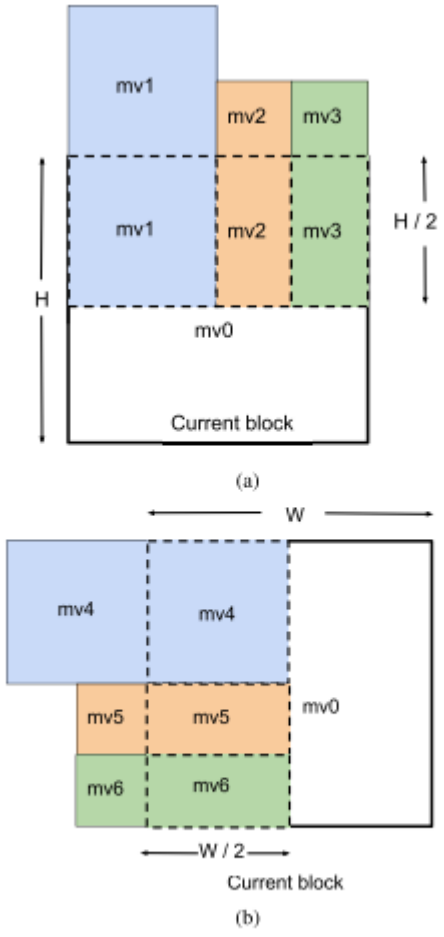


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

https://ieeexplore.ieee.org/document/8296419

	Motion vector coding accounts for a sizable portion of the overall bit rate. Modern video codecs typically adopt predictive coding for motion vectors and code the difference using entropy coding [30], [31]. The prediction accuracy has a large impact on the coding efficiency. AV1 employs a dynamic motion vector referencing scheme that obtains candidate motion vectors from the spatial and temporal neighbors and ranks them for efficient entropy coding. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7A4] defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data, and	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program which defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec, which is described above with respect to [10A3] and is incorporated here by reference. Specifically, a boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (*i.e.*, immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (*i.e.*, motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7A5] generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing the predictive picture;	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program which generates an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing the predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A practices overlapped block motion compensation using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (*i.e.*, prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (*i.e.*, that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

260

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

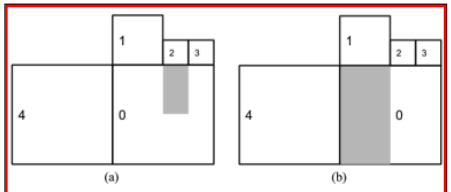


Fig. 1. Overlapping regions

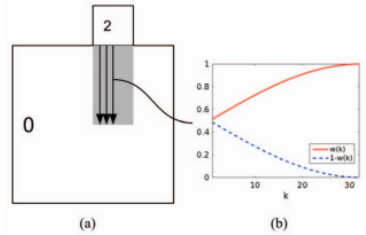


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

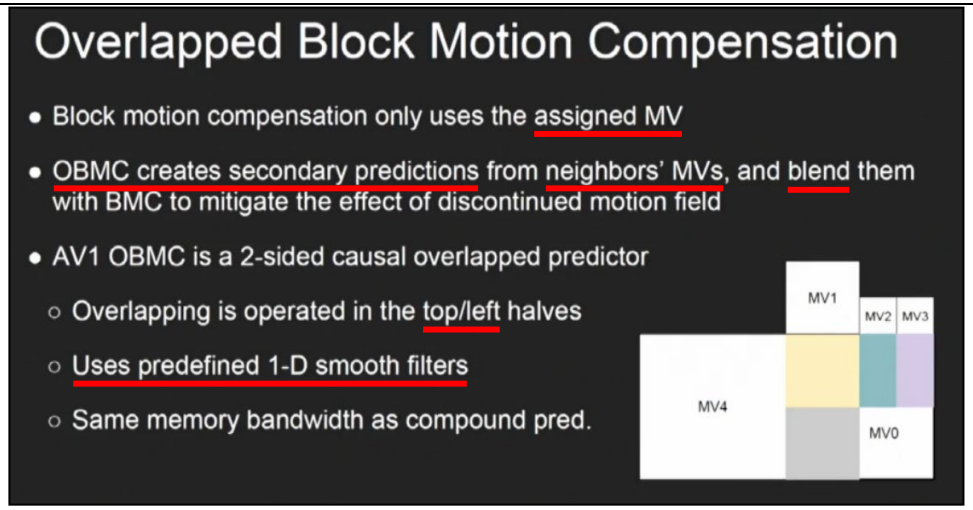
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

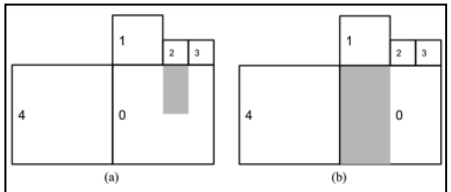


Fig. 1. Overlapping regions

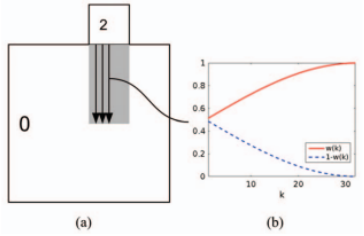


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

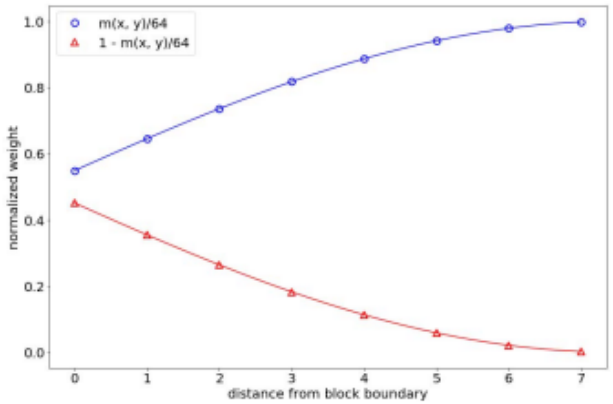


Fig. 15. ***Normalized weights for OBMC with H = 16 or W = 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7B] an entropy decoding program code to perform entropy decoding to the data thus demultiplexed to generate, at least, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure;	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising an entropy decoding program code to perform entropy decoding to the data thus demultiplexed to generate, at least, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory entropy decoding computer program code to program performs entropy decoding to the demultiplexed data required for constructing a specific syntax structure. Entropy decoding in AV1 generates, at least, the post-quantization data (*i.e.*, quantized transform coefficients), the border motion vector data (*i.e.*, at least the "the motion vector difference [that] will be entropy coded"), and parameter data (*i.e.*, quantization parameters and other "lateral data") for constructing a specific syntax structure (*i.e.*, open bitstream units). Entropy decoding is performed in order to construct a specific syntax structure (*i.e.*, "to compress the syntax elements"). In other words, the "output of this [entropy decoding] process is the variable symbol, containing a decoded syntax element."

employed for transform blocks at dimension 32×32 and above.

F. Quantization

The transform coefficients are quantized, and the quantization indexes are entropy coded. The quantization parameter (QP) in AV1 ranges between 0 and 255.

1) Quantization Step Size: At a given QP, the quantization step size for the dc coefficient is smaller than that for the ac coefficient. The mapping from QP to quantization step size for both dc and ac coefficients is drawn in Fig. 25. The lossless coding mode is achieved when QP is 0. By default, all the ac coefficients will use the same quantization step size. Since the human visual system tends to have different tolerance to distortions at various frequencies, AV1 also supports 15 sets of predefined quantization weighting matrices, where the quantization step size for each individual frequency component is further scaled differently. Each frame can optionally select a quantization weighting matrix set for luma and chroma planes, respectively.

2) Quantization Parameter Modulation: AV1 assigns a base QP for a coded frame, denoted by QP_{base}. The QP values for the dc and ac coefficients in both luma and chroma components are shown in Table 1. $\Delta QP_{p,b}$ are additional offset values transmitted in the frame header,

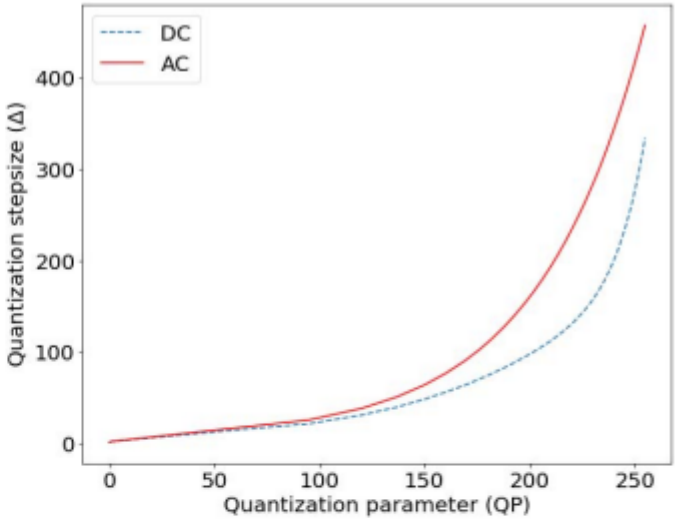


Fig. 25. ***QP and quantization step size maps for dc and ac coefficients.***

where $p \in \{Y, U, V\}$ denotes the plane and $b \in \{dc, ac\}$ denotes the dc or the ac transform coefficients.

Recognizing that the coding blocks within a frame may have different rate-distortion tradeoffs, AV1 further allows QP offset at both superblock and coding block levels. The resolution of QP offset at superblock level is signaled by the frame header. The available options are 1, 2, 4, and 8. The coding block-level QP offset can be achieved through

Table 1 Frame-Level QP Values (QP_{frame}) for Y/U/V Planes

	AC	DC
Y	QP_{base}	$QP_{base} + \Delta QP_{Y,DC}$
U	$QP_{base} + \Delta QP_{U,AC}$	$QP_{base} + \Delta QP_{U,DC}$
V	$QP_{base} + \Delta QP_{V,AC}$	$QP_{base} + \Delta QP_{V,DC}$

segmentation. AV1 allows a frame to classify its coding blocks into up to eight segments; each has its own QP offset decided by the frame header. The segment index associated with each coding block is sent through the bitstream to the decoder.

Therefore, the effective QP for ac coefficients in a coding block, QP_{cb}, is given by

$$QP_{cb} = \text{clip}(QP_{\text{frame}} + \Delta QP_{sb} + \Delta QP_{\text{seg}}, 1, 255) \quad (27)$$

where ΔQP_{sb} and ΔQP_{seg} are the QP offsets from the superblock and the segment, respectively. The clip function ensures it stays within a valid range. The QP is not allowed to change from a nonzero value to zero since zero is reserved for lossless coding.

The decoder rebuilds the quantized samples using a uniform quantizer. Given the quantization step size Δ and the quantization index k, the reconstructed sample is $k\Delta$.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The first four motion vectors in this ranked list will be used as candidate motion vector predictors. The encoder will pick the one that is closest to the desired motion vector and send its index to the decoder. It is not uncommon for coding blocks to have fewer than four candidate motion vectors, due to either the high flexibility in the reference frame selection, or a highly consistent motion activity in the local region. In such a context, the candidate motion vector list will be shorter than 4, which allows the codec to save bits spent on identifying the selected index. The dynamic candidate motion vector list is in contrast to the design in VP9, where one always constructs two candidate motion vectors. If not enough candidates are found, the VP9 codec will fill the list with zero vectors. AV1 also supports a special intermode that makes the inter-predictor use the frame-level affine model, as discussed in Section V-C2.

The motion vector difference will be entropy coded. Since a significant portion of the coding blocks will find a zero motion vector difference, the probability model is designed to account for such bias. AV1 allows a coding block to use 1 bit to indicate whether to directly use the selected motion vector predictor as its final motion vector or to additionally code the difference. The probability model for this entropy-coded bit is conditioned on two factors: whether its spatial neighbors have a nonzero motion vector difference and whether a sufficient number of motion vector predictors are found. For compound modes, where two motion vectors need to be specified, this extends to four cases that cover where either block, both, or neither one have a zero difference motion vector. The nonzero difference motion vector coding is consistent in all cases.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

276

VI. ENTROPY CODING SYSTEM

AV1 employs an M-ary symbol arithmetic coding method that was originally developed for the Daala video codec [39] to compress the syntax elements, where integer $M \in [2, 14]$. The probability model is updated per symbol coding.

C. Level Map Transform Coefficient Coding System

The transform coefficient entropy coding system is an intricate and performance-critical component in video codecs. We discuss its design in AV1 that decomposes it into a series of symbol codings.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

E. ENTROPY CODING

The entropy coding processes the symbols (quantized coefficients and lateral data) to reduce their statistical redundancy by applying lossless algorithms.

AV1 uses a symbol-to-symbol adaptive multi-symbol arithmetic coder with the probability being updated every new symbol. Each syntax element in AV1 is a member of an alphabet of N elements, and a context consists of a set of N probabilities together with a small count to facilitate fast early adaptation [2].

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

Quantization parameter
A variable used for scaling the quantized coefficients in the decoding process.

Quantized coefficient
A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

D. TRANSFORMS AND QUANTIZATION

The prediction error, or the residues, between the intra and inter prediction and the original blocks are processed by the transform module (T module, in Fig. 1), which converts the values from the spatial domain to the frequency domain. Then, the quantization step (Q module in Fig. 1) is applied to the transformed coefficients to attenuate or eliminate values associated with spectral components that are not perceptually relevant for the human visual system.

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

8.2. Parsing process for symbol decoder

8.2.1. General

The entropy decoder is referred to as the “Symbol decoder” and the functions init_symbol(sz), exit_symbol( ), read_symbol(cdf), and read_bool( ) are used in this Specification to indicate the entropy decoding operation.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

	8.2.2. Initialization process for symbol decoder The input to this process is a variable sz specifying the number of bytes to be read by the Symbol decoder. This process is invoked when the function init_symbol(sz) is called from the syntax structure. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

8.2.4. Exit process for symbol decoder

This process is invoked when the function exit_symbol( ) is called from the syntax structure.

It is a requirement of bitstream conformance that SymbolMaxBits is greater than or equal to -14 whenever this process is invoked.

The variable trailingBitPosition is set equal to get_position() - Min(15, SymbolMaxBits+15).

The bitstream position indicator is advanced by Max(0,SymbolMaxBits). (This skips over any trailing bits that have not already been read during symbol decode.)

The variable paddingEndPosition is set equal to get_position().

> **Note:** paddingEndPosition will always be a multiple of 8 indicating that the bit position is byte aligned.

It is a requirement of bitstream conformance that the bit at position trailingBitPosition is equal to 1.

It is a requirement of bitstream conformance that the bit at position x is equal to 0 for values of x strictly between trailingBitPosition and paddingEndPosition.

> **Note:** This exit process consumes the OBU trailing bits for a tile group.

If disable_frame_end_update_cdf is equal to 0 and TileNum is equal to context_update_tile_id, a copy is made of the final CDF values for each of the CDF arrays mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs. The name of the destination for the copy is the name of the CDF array prefixed with “Saved”. The name of the source for the copy is the name of the CDF array prefixed with “Tile”. For example, an array SavedYModeCdf will be created with values equal to TileYModeCdf.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 358-359

8.2.6. Symbol decoding process

The input to this process is an array cdf of length N + 1 which specifies the cumulative distribution for a symbol with N possible values.

The output of this process is the variable symbol, containing a decoded syntax element. The process also modifies the input array cdf to adapt the probabilities to the content of the stream.

This process is invoked when the function read_symbol(cdf) is called.

> **Note:** When this process is invoked, N will be greater than 1 and cdf[N-1] will be equal to 1 << 15.

The variables cur, prev, and symbol are calculated as follows:

```
cur = SymbolRange
symbol = -1
do {
  symbol++
  prev = cur
  f = ( 1 << 15 ) - cdf[ symbol ]
  cur = ((SymbolRange >> 8) * (f >> EC_PROB_SHIFT)) >> (7 - EC_PROB_SHIFT)
  cur += EC_MIN_PROB * (N - symbol - 1)
} while ( SymbolValue < cur )
```

Note: Implementations may prefer to store the inverse cdf to move the subtraction out of this loop.

The variable SymbolRange is set to prev - cur.

The variable SymbolValue is set equal to SymbolValue - cur.

The range and value are renormalized by the following ordered steps:

1. The variable bits is set to 15 – FloorLog2(SymbolRange). This represents the number of new bits to be added to SymbolValue.
2. The variable SymbolRange is set to SymbolRange << bits.
3. The variable numBits is set equal to Min(bits, Max(0, SymbolMaxBits)). This represents the number of new bits to read from the bitstream.
4. The variable newData is read using the f(numBits) parsing process.
5. The variable paddedData is set equal to newData << (bits - numBits).
6. The variable SymbolValue is set to paddedData ^ (((SymbolValue + 1) << bits) - 1).
7. The variable SymbolMaxBits is set to SymbolMaxBits - bits.

Note: bits may be equal to 0, in which case these ordered steps have no effect.

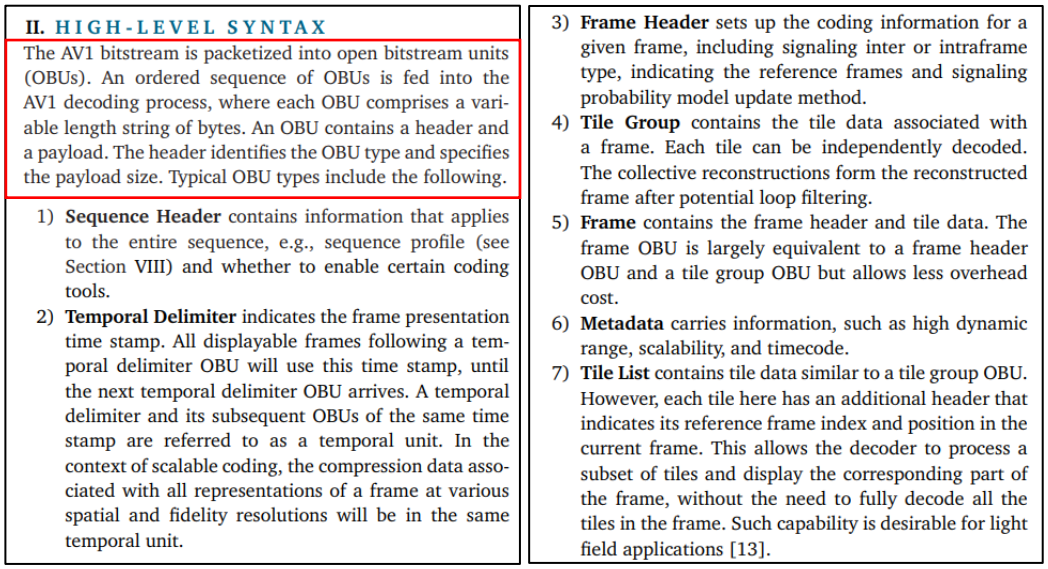
If disable_cdf_update is equal to 0, the cumulative distribution is updated as follows:

```
rate = 3 + ( cdf[ N ] > 15 ) + ( cdf[ N ] > 31 ) + Min( FloorLog2( N ), 2 )
tmp = 0
for ( i = 0; i < N - 1; i++ ) {
  tmp = ( i == symbol ) ? ( 1 << 15 ) : tmp
  if ( tmp < cdf[ i ] ) {
    cdf[ i ] -= ( ( cdf[ i ] - tmp ) >> rate )
  } else {
    cdf[ i ] += ( ( tmp - cdf[ i ] ) >> rate )
  }
}
cdf[ N ] += ( cdf[ N ] < 32 )
```

Note: The last entry of the cdf array is used to keep a count of the number of times the symbol has been decoded (up to a maximum of 32). This allows the cdf adaption rate to depend on the number of times the symbol has been decoded.

The return value from the function is given by symbol.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 359-361



II. HIGH-LEVEL SYNTAX

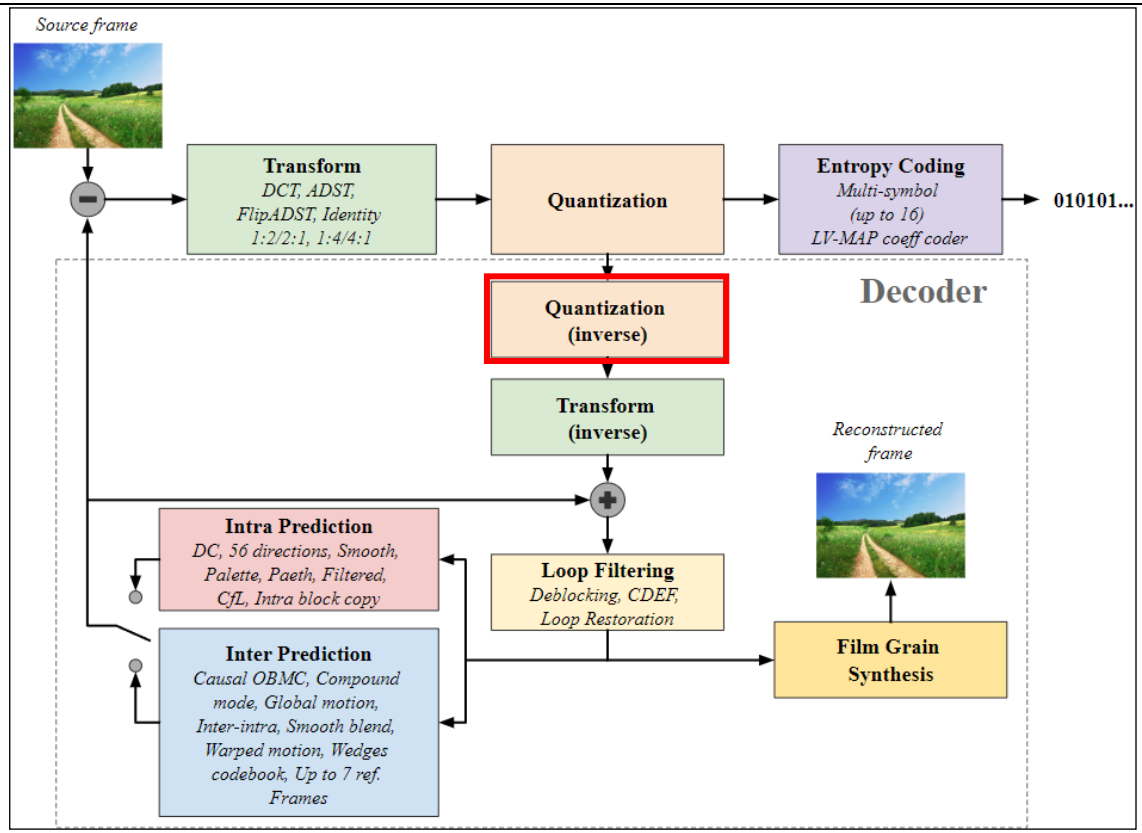
The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7C] an inverse-quantization program code to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data;	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising an inverse-quantization program code to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory inverse-quantization computer program code to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data.



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Dequantization

The process in which transform coefficients are obtained by scaling the quantized coefficients.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

II. AV1 AND VVC FEATURES

Both the AV1 and VVC encoders follow a hybrid block-based video coding scheme, which is based on the following signal and data processing operations, as presented in Fig. 1: (i) inter and intra-frame prediction, (ii) transform (T), (iii) quantization (Q), and (iv) entropy coding [11]. A reconstruction loop with inverse quantization (IQ) and inverse transform (IT) is also included to guarantee that the encoder and decoder will use the same reference data [11]. Finally,

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12. Reconstruction and dequantization

7.12.1. General

This section details the process of reconstructing a block of coefficients using dequantization and inverse transforms.

7.12.2. Dequantization functions

This section defines the functions get_dc_quant and get_ac_quant that are needed by the dequantization process.

The quantization parameters are derived from lookup tables.

The function get_dc_quant(plane) returns the quantizer value for the dc coefficient for a particular plane and is derived as follows:

- If plane is equal to 0, return dc_q(get_qindex(0, segment_id) + DeltaQYDc).
- Otherwise if plane is equal to 1, return dc_q(get_qindex(0, segment_id) + DeltaQUDc).
- Otherwise (plane is equal to 2), return dc_q(get_qindex(0, segment_id) + DeltaQVDc).

The function get_ac_quant(plane) returns the quantizer value for the ac coefficient for a particular plane and is derived as follows:

- If plane is equal to 0, return ac_q(get_qindex(0, segment_id)).
- Otherwise if plane is equal to 1, return ac_q(get_qindex(0, segment_id) + DeltaQUAc).
- Otherwise (plane is equal to 2), return ac_q(get_qindex(0, segment_id) + DeltaQVAc).

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-294

7.13. Inverse transform process

7.13.1. General

This section details the inverse transforms used during the reconstruction processes detailed in section 7.12.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 296

The generated post-quantization transform coefficients data are orthogonal because at least some of the inverse transform processes are orthogonal (*e.g.*, Inverse Discrete Cosine Transform (DCT) process).

7.13.2.2. Inverse DCT array permutation process

This process performs an in-place permutation of the array T of length 2^n for $2 \le n \le 6$ which is required before execution of the inverse DCT process.

The input to this process is a variable n that specifies the base 2 logarithm of the length of the input array.

A temporary array named copyT is set equal to T.

T[i] is set equal to copyT[brev(n, i)] for i = 0..((1 << n) - 1).

7.13.2.3. Inverse DCT process

This process performs an in-place inverse discrete cosine transform of the permuted array T which is of length 2^n for $2 \le n \le 6$.

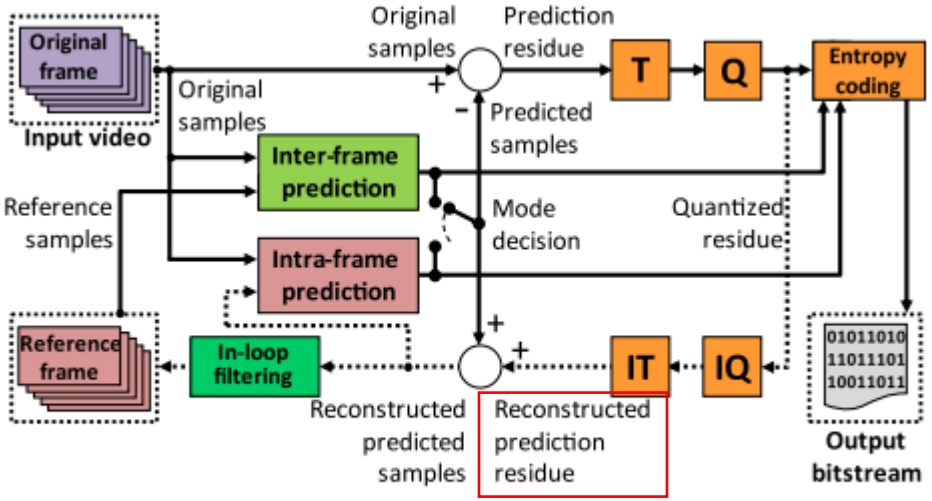
The inputs to this process are:

- a variable n that specifies the base 2 logarithm of the length of the input array,
- a variable r that specifies the intermediate clamping range.

The following ordered steps apply:

1. Invoke the inverse DCT permutation process as specified in section 7.13.2.2 with the input variable n.
2. If n is equal to 6, invoke B(32 + i, 63 - i, 63 - 4 * brev(4, i), 0, r) for i = 0..15.
3. If n is greater than or equal to 5, invoke B(16 + i, 31 - i, 6 + (brev(3, 7 - i) << 3), 0, r) for i = 0..7.
4. If n is equal to 6, invoke H(32 + i * 2, 33 + i * 2, i & 1, r) for i = 0..15.
5. If n is greater than or equal to 4, invoke B(8 + i, 15 - i, 12 + (brev(2, 3 - i) << 4), 0, r) for i = 0..3.
6. If n is greater than or equal to 5, invoke H(16 + 2 * i, 17 + 2 * i, i & 1, r) for i = 0..7.
7. If n is equal to 6, invoke B(62 - i * 4 - j, 33 + i * 4 + j, 60 - 16 * brev(2, i) + 64 * j, 1, r) for i = 0..3, for j = 0..1.
8. If n is greater than or equal to 3, invoke B(4 + i, 7 - i, 56 - 32 * i, 0, r) for i = 0..1.
9. If n is greater than or equal to 4, invoke H(8 + 2 * i, 9 + 2 * i, i & 1, r) for i = 0..3.
10. If n is greater than or equal to 5, invoke B(30 - 4 * i - j, 17 + 4 * i + j, 24 + (j << 6) + ((1 - i) << 5), 1, r) for i = 0..1, for j=0..1.
11. If n is equal to 6, invoke H(32 + i * 4 + j, 35 + i * 4 - j, i & 1, r) for i = 0..7, for j = 0..1.
12. Invoke B(2 * i, 2 * i + 1, 32 + 16 * i, 1 - i, r) for i = 0..1.
13. If n is greater than or equal to 3, invoke H(4 + 2 * i, 5 + 2 * i, i, r) for i = 0..1.
14. If n is greater than or equal to 4, invoke B(14 - i, 9 + i, 48 + 64 * i, 1, r) for i = 0..1.
15. If n is greater than or equal to 5, invoke H(16 + 4 * i + j, 19 + 4 * i - j, i & 1, r) for i = 0..3, for j = 0..1.
16. If n is equal to 6, invoke B(61 - i * 8 - j, 34 + i * 8 + j, 56 - i * 32 + (j >> 1) * 64, 1, r) for i = 0..1, for j = 0..3.
17. Invoke H(i, 3 - i, 0, r) for i = 0..1.
18. If n is greater than or equal to 3, invoke B(6, 5, 32, 1, r).
19. If n is greater than or equal to 4, invoke H(8 + 4 * i + j, 11 + 4 * i - j, i, r) for i = 0..1, for j = 0..1.
20. If n is greater than or equal to 5, invoke B(29 - i, 18 + i, 48 + (i >> 1) * 64, 1, r) for i = 0..3.
21. If n is equal to 6, invoke H(32 + 8 * i + j, 39 + 8 * i - j, i & 1, r) for i = 0..3, for j = 0..3.
22. If n is greater than or equal to 3, invoke H(i, 7 - i, 0, r) for i = 0..3.

	23. If n is greater than or equal to 4, invoke B(13 - i, 10 + i, 32, 1, r) for i = 0..1. 24. If n is greater than or equal to 5, invoke H(16 + i * 8 + j, 23 + i * 8 - j, i, r) for i = 0..1, for j = 0..3. 25. If n is equal to 6, invoke B(59 - i, 36 + i, i < 4 ? 48 : 112, 1, r) for i = 0..7. 26. If n is greater than or equal to 4, invoke H(i, 15 - i, 0, r) for i = 0..7. 27. If n is greater than or equal to 5, invoke B(27 - i, 20 + i, 32, 1, r) for i = 0..3. 28. If n is equal to 6, the following steps apply for i = 0..7: ◦ Invoke H(32 + i, 47 - i, 0, r). ◦ Invoke H(48 + i, 63 - i, 1, r). 29. If n is greater than or equal to 5, invoke H(i, 31 - i, 0, r) for i = 0..15. 30. If n is equal to 6, invoke B(55 - i, 40 + i, 32, 1, r) for i = 0..7. 31. If n is equal to 6, invoke H(i, 63 - i, 0, r) for i = 0..31. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 298-300 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7D] an inverse-orthogonal transform program code to perform inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area;	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising: an inverse-orthogonal transform program code to perform inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A's AV1 codec uses a non-transitory computer program code to perform an inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data (as described above with respect to [7C], which is incorporated here by reference) to produce a decoded residual picture of one video area (*i.e.*, residual or reconstructed prediction residue).  **FIGURE 1. Block diagram of a typical hybrid video encoder.** https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

The reconstruction and dequantization process is defined as follows:

The variable dqDenom is derived as follows:

- If txSz is equal to TX_32X32, TX_16X32, TX_32X16, TX_16X64, or TX_64X16, dqDenom is set equal to 2.

- Otherwise, if txSz is equal to TX_64X64, TX_32X64, or TX_64X32, dqDenom is set equal to 4.
- Otherwise, dqDenom is set equal to 1.

The variable log2W (specifying the base 2 logarithm of the width of the transform block) is set equal to Tx_Width_Log2[txSz].

The variable log2H (specifying the base 2 logarithm of the height of the transform block) is set equal to Tx_Height_Log2[txSz].

The variable w (specifying the width of the transform block) is set equal to 1 << log2W.

The variable h (specifying the height of the transform block) is set equal to 1 << log2H.

The variable tw is set equal to Min(32, w).

The variable th is set equal to Min(32, h).

The variable flipUD is derived as follows. If PlaneTxType is equal to one of FLIPADST_DCT, FLIPADST_ADST, V_FLIPADST, or FLIPADST_FLIPADST, flipUD is set equal to 1. Otherwise, flipUD is set equal to 0.

The variable flipLR is derived as follows. If PlaneTxType is equal to one of DCT_FLIPADST, ADST_FLIPADST, H_FLIPADST, or FLIPADST_FLIPADST, flipLR is set equal to 1. Otherwise, flipLR is set equal to 0.

The following ordered steps apply:

1. For i = 0..(th-1), for j = 0..(tw-1), the following ordered steps apply:

   a. The variable q is derived as follows:

      - If i is equal to 0 and j is equal to 0, the variable q is set equal to get_dc_quant(plane).
      - Otherwise (i, j or both are not equal to 0), the variable q is set equal to get_ac_quant(plane).

   b. The variable q2 is derived as follows:

      - If using_qmatrix is equal to 1, PlaneTxType is less than IDTX, and SegQMLevel[plane][segment_id] is less than 15, q2 is set equal to Round2(q * Quantizer_Matrix[SegQMLevel[plane][segment_id]][plane > 0][Qm_Offset[txSz] + i * tw + j], 5).
      - Otherwise, q2 is set equal to q.

   c. The variable dq is set equal to Quant[i * tw + j] * q2.

   d. The variable sign is set equal to (dq < 0) ? -1 : 1.

   e. The variable dq2 is set equal to sign * (Abs(dq) & 0xFFFFFF) / dqDenom.

   f. Dequant[i][j] is set equal to Clip3(- (1 << (7 + BitDepth)), (1 << (7 + BitDepth)) - 1, dq2).

2. Invoke the 2D inverse transform block process defined in section 7.13.3 with the variable txSz as input. The inverse transform outputs are stored in the Residual buffer.

3. For i = 0..(h-1), for j = 0..(w-1), the following applies:

    - The variable xx is set equal to flipLR ? (w - j - 1) : j.
    - The variable yy is set equal to flipUD ? (h - i - 1) : i.
    - CurrFrame[plane][y + yy][x + xx] is set equal to Clip1(CurrFrame[plane][y + yy][x + xx] + Residual[i][j]).

If Lossless is equal to 1, it is a requirement of bitstream conformance that the values written into the Residual array in step 2 are representable by a signed integer with 1 + BitDepth bits.

Note: When Lossless is equal to 0, values written into Residual may not be representable by 1 + BitDepth bits (for example, due to quantization noise). The constraints in other parts of the specification ensure that the values will always be representable by a signed integer with Max(BitDepth + 5, 15) bits.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-296

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

	The input to this process is a variable txSz that specifies the transform size. Set the variable log2W equal to Tx_Width_Log2[txSz]. Set the variable log2H equal to Tx_Height_Log2[txSz]. Set the variable w equal to 1 << log2W. Set the variable h equal to 1 << log2H.

Set the variable rowShift equal to Lossless ? 0 : Transform_Row_Shift[txSz].

Set the variable colShift equal to Lossless ? 0 : 4.

Set the variable rowClampRange equal to BitDepth + 8.

Set the variable colClampRange equal to Max(BitDepth + 6, 16).

The row transforms with i = 0..(h-1) are applied as follows:

- T[j] is derived as follows for j = 0..(w-1):
    - If i and j are both less than 32, T[j] is set equal to Dequant[i][j].
    - Otherwise, T[j] is set equal to 0.
- If Abs(log2W - log2H) is equal to 1, T[j] is set equal to Round2(T[j] * 2896, 12) for j = 0..(w-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 2.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, ADST_DCT, FLIPADST_DCT or H_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, if PlaneTxType is equal to one of DCT_ADST, ADST_ADST, DCT_FLIPADST, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, H_ADST, or H_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2W.
- Set Residual[i][j] equal to Round2(T[j], rowShift) for j = 0..(w-1).

Between the row and column transforms, Residual[i][j] is set equal to Clip3(- (1 << (colClampRange - 1)), (1 << (colClampRange - 1)) - 1, Residual[i][j]) for i = 0..(h-1), for j = 0..(w-1).

The column transforms with j = 0..(w-1) are applied as follows:

- Set T[i] equal to Residual[i][j] for i = 0..(h-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 0.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, DCT_ADST, DCT_FLIPADST or V_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2H and the input variable r equal to colClampRange.

- Otherwise, if PlaneTxType is equal to one of ADST_DCT, ADST_ADST, FLIPADST_DCT, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, V_ADST, or V_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2H and the input variable r equal to colClampRange.

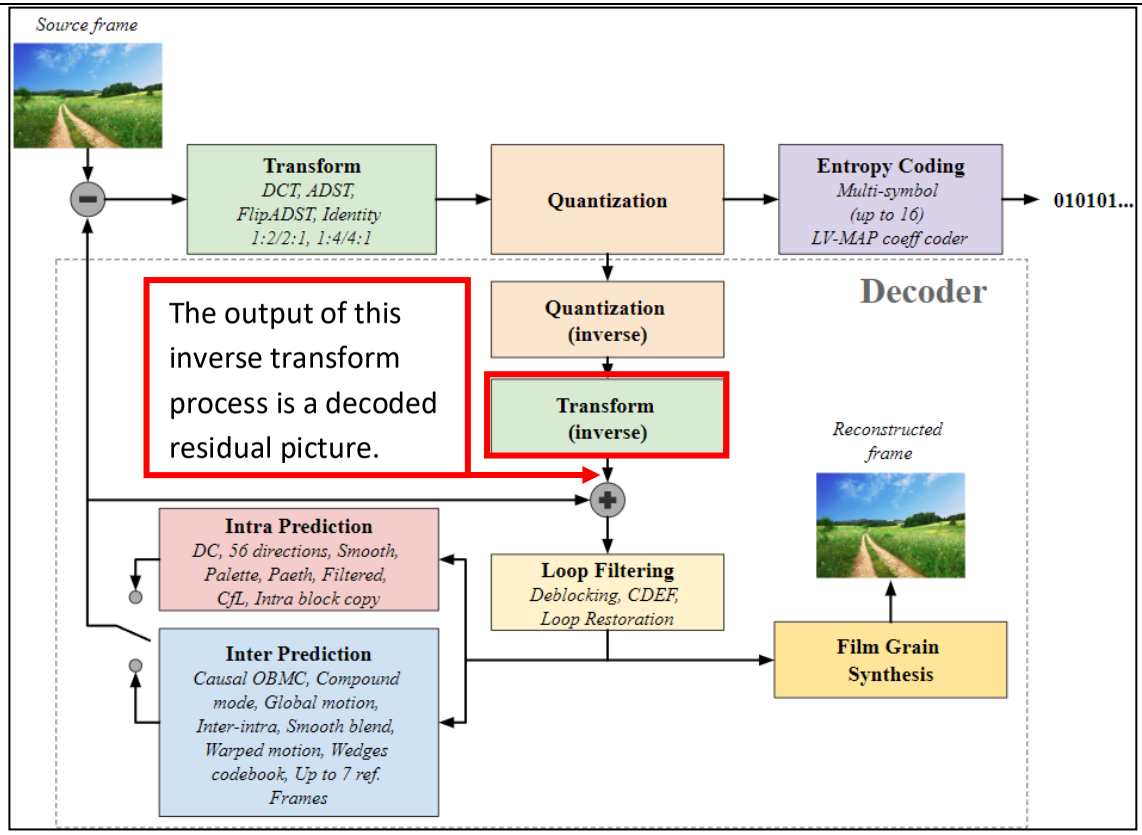
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2H.

- Residual[i][j] is set equal to Round2(T[i], colShift) for i = 0..(h-1).

where Transform_Row_Shift is defined as:

```
Transform_Row_Shift[ TX_SIZES_ALL ] = {
  0, 1, 2, 2, 2, 0, 0, 1, 1,
  1, 1, 1, 1, 1, 1, 2, 2, 2, 2
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305-307



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7E1] a zone-border motion compensation program code to define a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data, and	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising a zone-border motion compensation program code to define a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* elements [7A3] and [7A4], which are incorporated here by reference. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program that comprises zone-border motion compensation program code to define a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec. A boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (*i.e.*, immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (*i.e.*, motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7E2] generate an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing a first predictive picture; and	The Samsung Accused Products are a non-transitory computer readable device having stored thereon a computer program for generating an estimated video signal in each rectangular zone in the picture to be decoded, that satisfies Poisson's Equation, thus producing a first predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* element [7A5], which is incorporated here by reference. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program to generate an estimated video signal using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (*i.e.*, prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (*i.e.*, that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

301

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

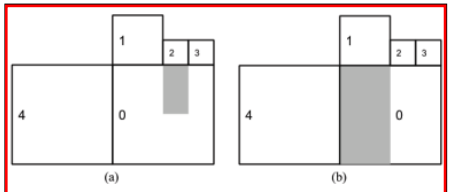


Fig. 1. Overlapping regions

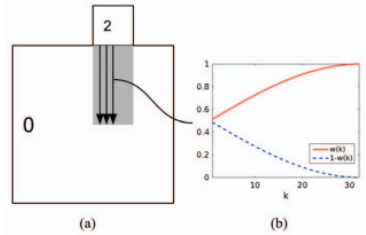


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

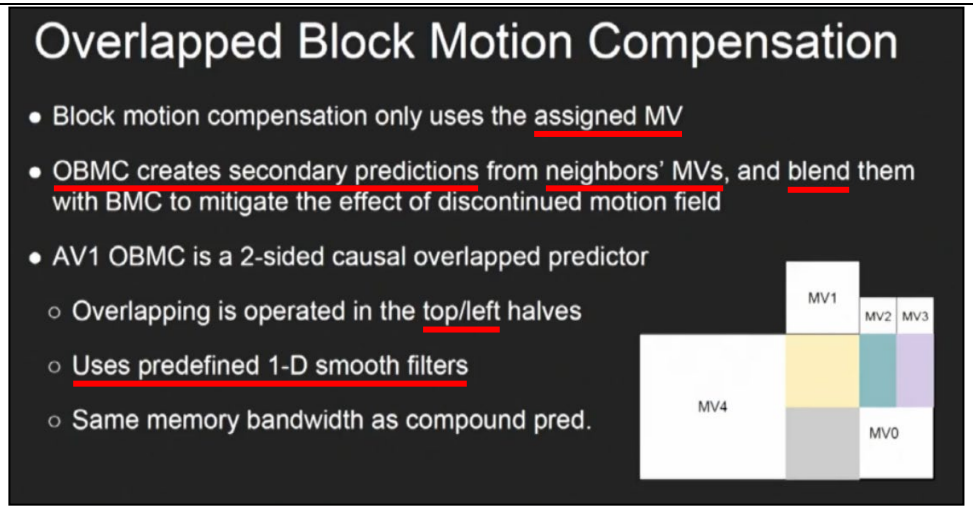
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

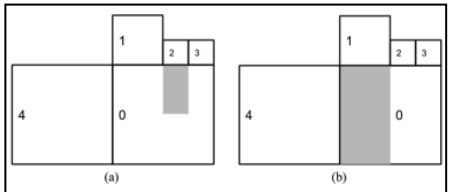


Fig. 1. Overlapping regions

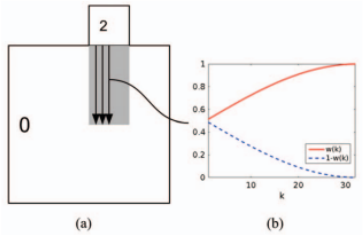


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

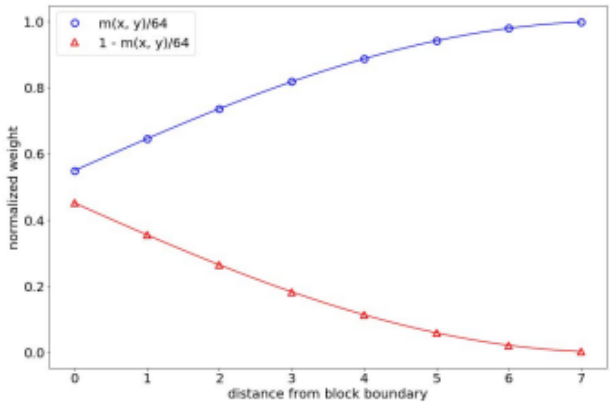


Fig. 15. ***Normalized weights for OBMC with H = 16 or W = 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

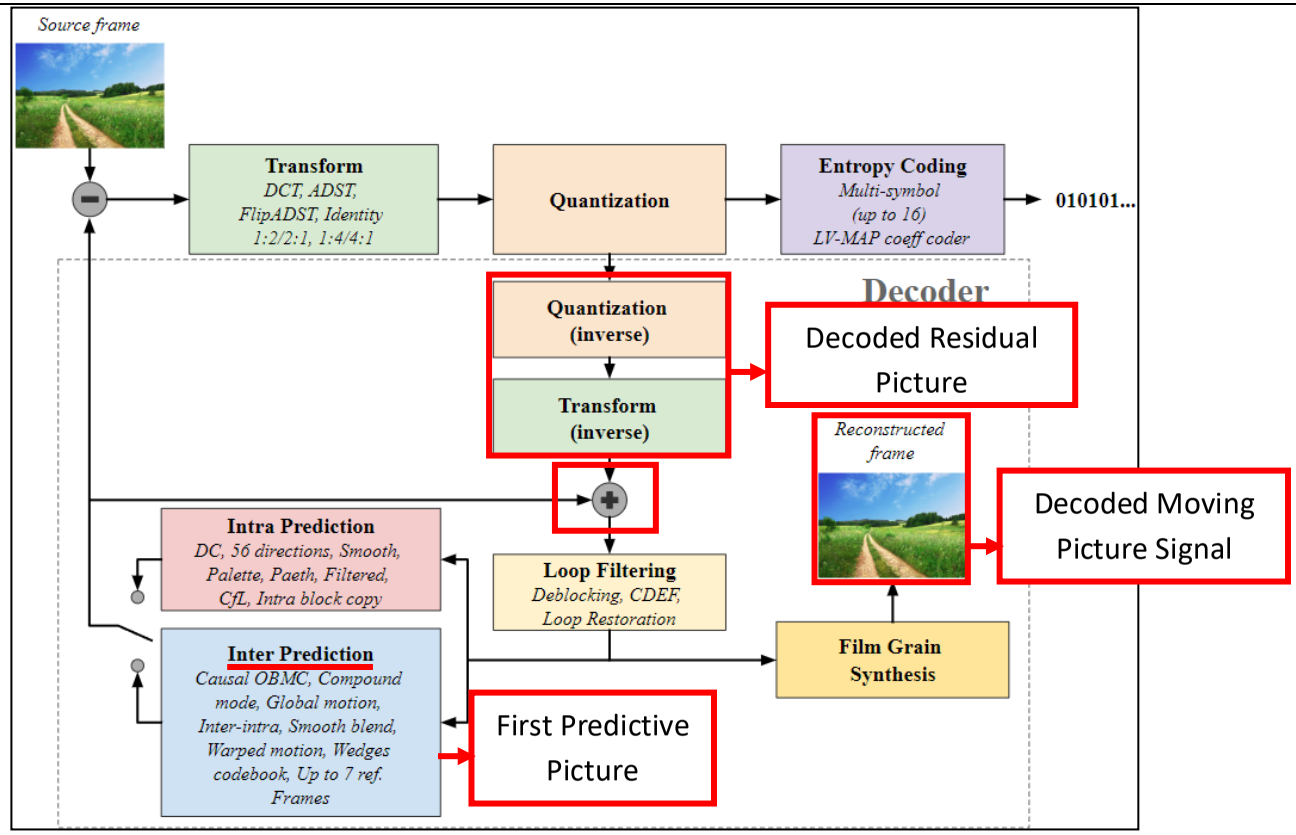
$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7F] a combine program code to combine the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal,	The Samsung Accused Products are non-transitory computer readable devices having stored thereon a computer program comprising a combine program code to combine the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program to combine the predictive picture and the decoded residual picture to generate a decoded moving-picture signal using the AV1 codec. Further, to generate a decoded moving picture signal (*i.e.*, the reconstructed frame or reconstructed predicted samples), the AV1 codec combines (*i.e.*, reconstructs) the first predictive picture (*e.g.*, the inter predicted picture or corresponding prediction values) and the decoded residual picture (*i.e.,* the decoded residual, the inverse quantized and inverse transformed picture, or reconstructed prediction residue). **Reconstruction** Obtaining the addition of the decoded residual and the corresponding prediction values. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

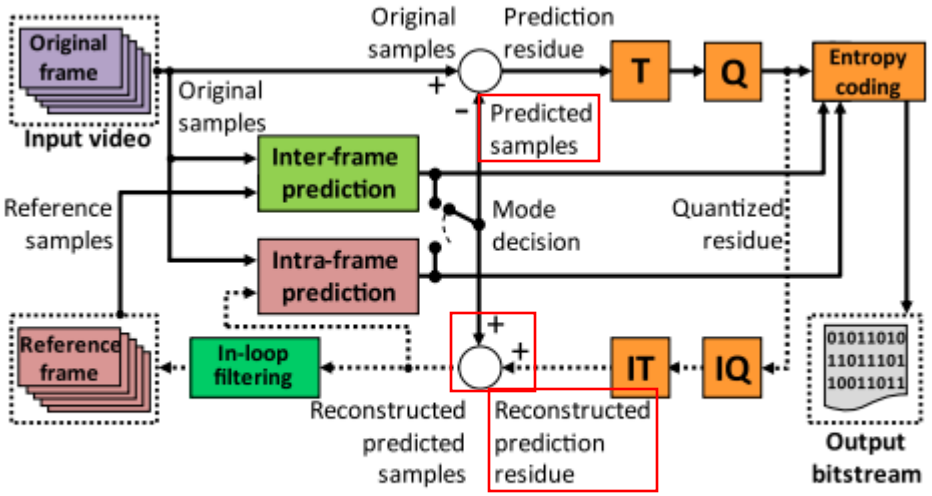


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

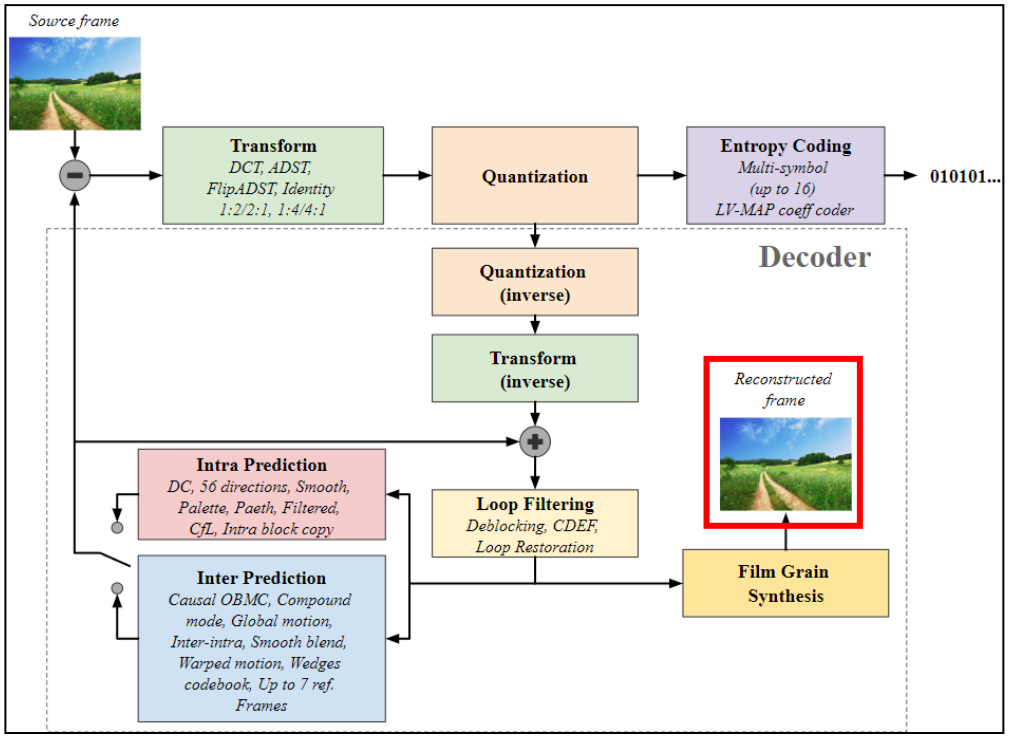
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[7G] the decoded moving-picture signal for at least one picture being stored as a reference picture.	The decoded moving-picture signal for at least one picture is stored as a reference picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec to store the decoded moving-picture signal (*i.e.*, decoded samples for the current frame) as reference frames (*i.e.*, an updated set of reference frames) and each of its parameters in memory.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.20. Reference frame update process

This process is invoked as the final step in decoding a frame.

The inputs to this process are the decoded samples for the current frame LrFrame[plane][x][y].

The output from this process is an updated set of reference frames and previous motion vectors.

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefValid[i] is set equal to 1.
- RefFrameId[i] is set equal to current_frame_id.
- RefUpscaledWidth[i] is set equal to UpscaledWidth.
- RefFrameWidth[i] is set equal to FrameWidth.
- RefFrameHeight[i] is set equal to FrameHeight.
- RefRenderWidth[i] is set equal to RenderWidth.
- RefRenderHeight[i] is set equal to RenderHeight.

	• RefMiCols[i] is set equal to MiCols. • RefMiRows[i] is set equal to MiRows. • RefFrameType[i] is set equal to frame_type. • RefSubsamplingX[i] is set equal to subsampling_x. • RefSubsamplingY[i] is set equal to subsampling_y. • RefBitDepth[i] is set equal to BitDepth. • SavedOrderHints[i][j + LAST_FRAME] is set equal to OrderHints[j + LAST_FRAME] for j = 0..REFS_PER_FRAME-1. • FrameStore[i][0][y][x] is set equal to LrFrame[0][y][x] for x = 0..UpscaledWidth-1, for y = 0..FrameHeight-1. • FrameStore[i][plane][y][x] is set equal to LrFrame[plane][y][x] for plane = 1..2, for x = 0..((UpscaledWidth + subsampling_x) >> subsampling_x) - 1, for y = 0..((FrameHeight + subsampling_y) >> subsampling_y) - 1. • SavedRefFrames[i][row][col] is set equal to MfRefFrames[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1. • SavedMvs[i][row][col][comp] is set equal to MfMvs[row][col][comp] for comp = 0..1, for row = 0..MiRows-1, for col = 0..MiCols-1. • SavedGmParams[i][ref][j] is set equal to gm_params[ref][j] for ref = LAST_FRAME..ALTREF_FRAME, for j = 0..5. • SavedSegmentIds[i][row][col] is set equal to SegmentIds[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1. • The function save_cdfs(i) is invoked (see below). • If film_grain_params_present is equal to 1, the function save_grain_params(i) is invoked (see below). • The function save_loop_filter_params(i) is invoked (see below). • The function save_segmentation_params(i) is invoked (see below).

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefOrderHint[i] is set equal to OrderHint.

save_cdfs(ctx) is a function call that indicates that all the CDF arrays are saved into frame context number ctx in the range 0 to (NUM_REF_FRAMES - 1). When this function is invoked the following takes place:

- A copy of each CDF array mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs is saved in an area of memory indexed by ctx.

save_grain_params(i) is a function call that indicates that all the syntax elements that can be read in film_grain_params should be saved into an area of memory indexed by i.

save_loop_filter_params(i) is a function call that indicates that the values of loop_filter_ref_deltas[j] for j = 0 .. TOTAL_REFS_PER_FRAME-1, and the values of loop_filter_mode_deltas[j] for j = 0 .. 1 should be saved into an area of memory indexed by i.

save_segmentation_params(i) is a function call that indicates that the values of FeatureEnabled[j][k] and FeatureData[j][k] for j = 0 .. MAX_SEGMENTS-1, for k = 0 .. SEG_LVL_MAX-1 should be saved into an area of memory indexed by i.

Note: Although this process stores all the motion vectors into SavedMvs, only the values where row and col are both odd will affect the decoding process.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 349-351

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[8P] The moving-picture decoding program according to claim 7 wherein the entropy decoding program code further generates motion-vector data, the program further comprises:	The entropy decoding program code of the Samsung Accused Products further generates motion-vector data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses the AV1 codec to further generate motion-vector data when performing entropy decoding. Specifically, the AV1 codec further generates motion-vector data including "adopt[ing] predictive coding for motion vectors and cod[ing] the difference [in the prediction and actual motion vectors] using entropy coding," and ranking "obtain[ed] candidate motion vectors from the spatial and temporal neighbors" for entropy coding. **D. Dynamic Motion Vector Referencing Scheme** Motion vector coding accounts for a sizable portion of the overall bit rate. Modern video codecs typically adopt predictive coding for motion vectors and code the difference using entropy coding [30], [31]. The prediction accuracy has a large impact on the coding efficiency. AV1 employs a dynamic motion vector referencing scheme that obtains candidate motion vectors from the spatial and temporal neighbors and ranks them for efficient entropy coding. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The first four motion vectors in this ranked list will be used as candidate motion vector predictors. The encoder will pick the one that is closest to the desired motion vector and send its index to the decoder. It is not uncommon for coding blocks to have fewer than four candidate motion vectors, due to either the high flexibility in the reference frame selection, or a highly consistent motion activity in the local region. In such a context, the candidate motion vector list will be shorter than 4, which allows the codec to save bits spent on identifying the selected index. The dynamic candidate motion vector list is in contrast to the design in VP9, where one always constructs two candidate motion vectors. If not enough candidates are found, the VP9 codec will fill the list with zero vectors. AV1 also supports a special intermode that makes the inter-predictor use the frame-level affine model, as discussed in Section V-C2.

The motion vector difference will be entropy coded. Since a significant portion of the coding blocks will find a zero motion vector difference, the probability model is designed to account for such bias. AV1 allows a coding block to use 1 bit to indicate whether to directly use the selected motion vector predictor as its final motion vector or to additionally code the difference. The probability model for this entropy-coded bit is conditioned on two factors: whether its spatial neighbors have a nonzero motion vector difference and whether a sufficient number of motion vector predictors are found. For compound modes, where two motion vectors need to be specified, this extends to four cases that cover where either block, both, or neither one have a zero difference motion vector. The nonzero difference motion vector coding is consistent in all cases.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	An entropy decoding program code further generates motion vector data (*e.g.*, the difference in the prediction and actual motion vectors using entropy coding) because it utilizes the entropy coded motion vector data described above as its input. 8.2. Parsing process for symbol decoder 8.2.1. General The entropy decoder is referred to as the "Symbol decoder" and the functions init_symbol(sz), exit_symbol( ), read_symbol(cdf), and read_bool( ) are used in this Specification to indicate the entropy decoding operation. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353 8.2.2. Initialization process for symbol decoder The input to this process is a variable sz specifying the number of bytes to be read by the Symbol decoder. This process is invoked when the function init_symbol(sz) is called from the syntax structure. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[8A] a motion compensation program code to specify a corresponding rectangular zone in the reference picture based on the motion-vector data, thus generating a second predictive picture;	The Samsung Accused Products comprise: a motion compensation program code to specify a corresponding rectangular zone in the reference picture based on the motion-vector data, thus generating a second predictive picture. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a motion compensation program code (*e.g.*, translational motion compensation) to specify a corresponding rectangular zone in the reference picture (*e.g.*, target reference block whose top pixel is located at (x_1, y_1)) based on the motion-vector data (*e.g.*, using the motion vector obtained through motion estimation), thus generating a second predictive picture (*e.g.*, the coding block's motion prediction). *C. Inter Frame Prediction* AV1 supports a rich toolsets to exploit the temporal correlation in video signals. These include adaptive filtering in translational motion compensation, affine motion compensation, and highly flexible compound prediction modes. *1) Translational Motion Compensation:* A coding block uses a motion vector to find its prediction in a reference frame. It first maps its current position, e.g. top-left pixel position (x_0, y_0) in Figure 7, in the reference frame. It is then displaced by the motion vector to the target reference block whose top-left pixel is located at (x_1, y_1).

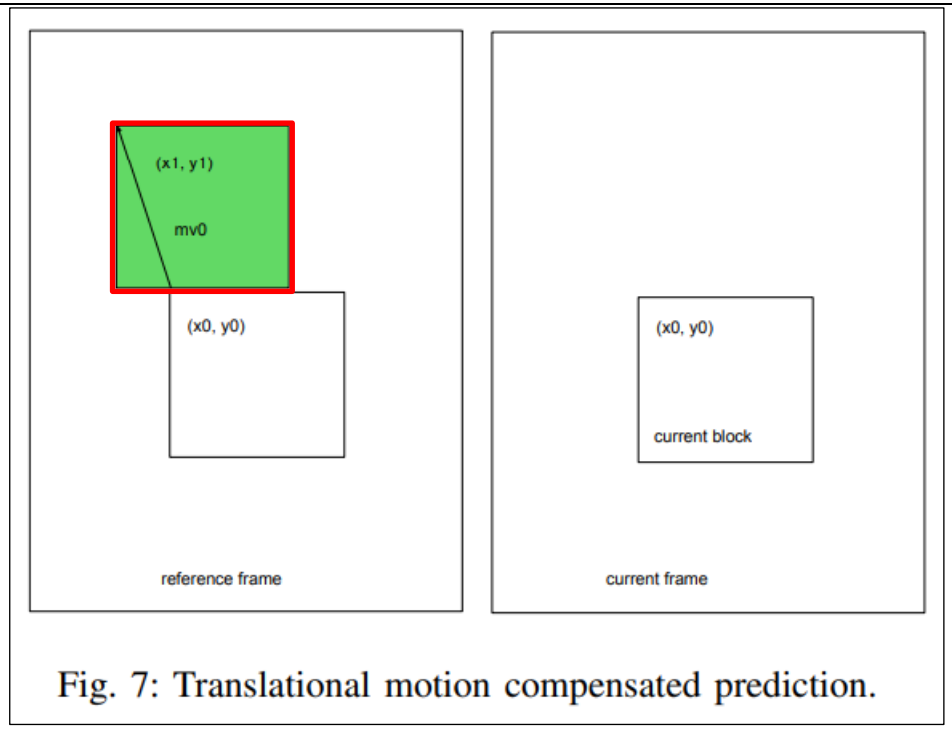


Fig. 7: Translational motion compensated prediction.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

<table>
<tr>
<td>[8B] a selection program code to select either the first predictive picture or the second predictive picture and supply the predictive picture thus selected to the combiner; and</td>
<td>The Samsung Accused Products comprise: a selection program code to select either the first predictive picture or the second predictive picture and supply the predictive picture thus selected to the combiner. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec.

This element is infringed literally, or in the alternative, under the doctrine of equivalents.

For example, the “motion-mode” flag is a selection program code to select either the first predictive picture (e.g., the predictive picture generated with OBMC) or the second predictive picture (e.g., predictive picture generated with translational motion compensation) and supply the predictive picture thus selected to the combiner.

<table>
<tr><td>MOTION_MODES</td><td>3</td><td>Number of values for motion modes</td></tr>
<tr><td>SIMPLE</td><td>0</td><td>Use translation or global motion compensation</td></tr>
<tr><td>OBMC</td><td>1</td><td>Use overlapped block motion compensation</td></tr>
<tr><td>LOCALWARP</td><td>2</td><td>Use local warp motion compensation</td></tr>
</table>
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 12

<table>
<tr><td>6.8.2. Uncompressed header semantics</td></tr>
<tr><td>is_motion_mode_switchable equal to 0 specifies that only the SIMPLE motion mode will be used.</td></tr>
</table>
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 154</td>
</tr>
</table>

6.10.26. Read motion mode semantics

use_obmc equal to 1 means that OBMC should be used. use_obmc equal to 0 means that simple translation should be used.

motion_mode specifies the type of motion compensation to perform:

motion_mode	Name of motion_mode
0	SIMPLE
1	OBMC
2	LOCALWARP

Note: A motion_mode equal to SIMPLE is used for blocks requiring global motion.

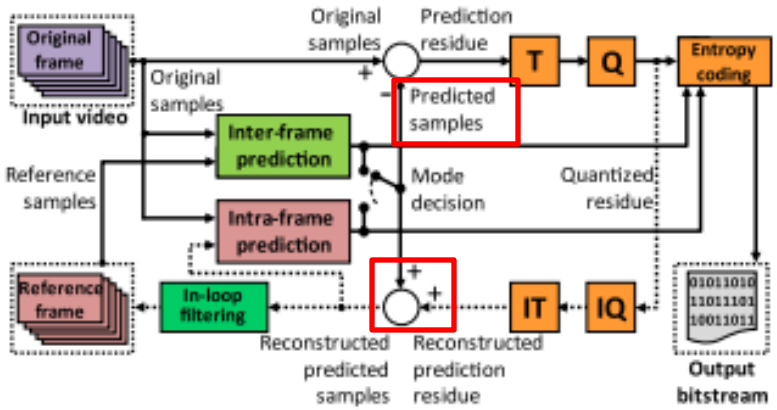
https://aomediacodec.github.io/av1-spec/av1-spec.pdf

5.11. Tile group OBU syntax

5.11.27. Read motion mode syntax

read_motion_mode(isCompound) {	Type
if (skip_mode) {	
motion_mode = SIMPLE	
return	
}	
if (!is_motion_mode_switchable) {	
motion_mode = SIMPLE	
return	
}	
if (Min(Block_Width[MiSize],	
Block_Height[MiSize]) < 8) {	
motion_mode = SIMPLE	
return	
}	
if (!force_integer_mv &&	
(YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV)) {	
if (GmType[RefFrame[0]] > TRANSLATION) {	
motion_mode = SIMPLE	
return	
}	
}	
if (isCompound || RefFrame[1] == INTRA_FRAME || !has_overlappable_candidates( )) {	
motion_mode = SIMPLE	
return	
}	
find_warp_samples()	
if (force_integer_mv || NumSamples == 0 ||	
!allow_warped_motion || is_scaled(RefFrame[0])) {	
use_obmc	S()
motion_mode = use_obmc ? OBMC : SIMPLE	
} else {	
motion_mode	S()

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	As shown below, the selected predictive picture is provided to the combiner.  FIGURE 1. Block diagram of a typical hybrid video encoder. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8= Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

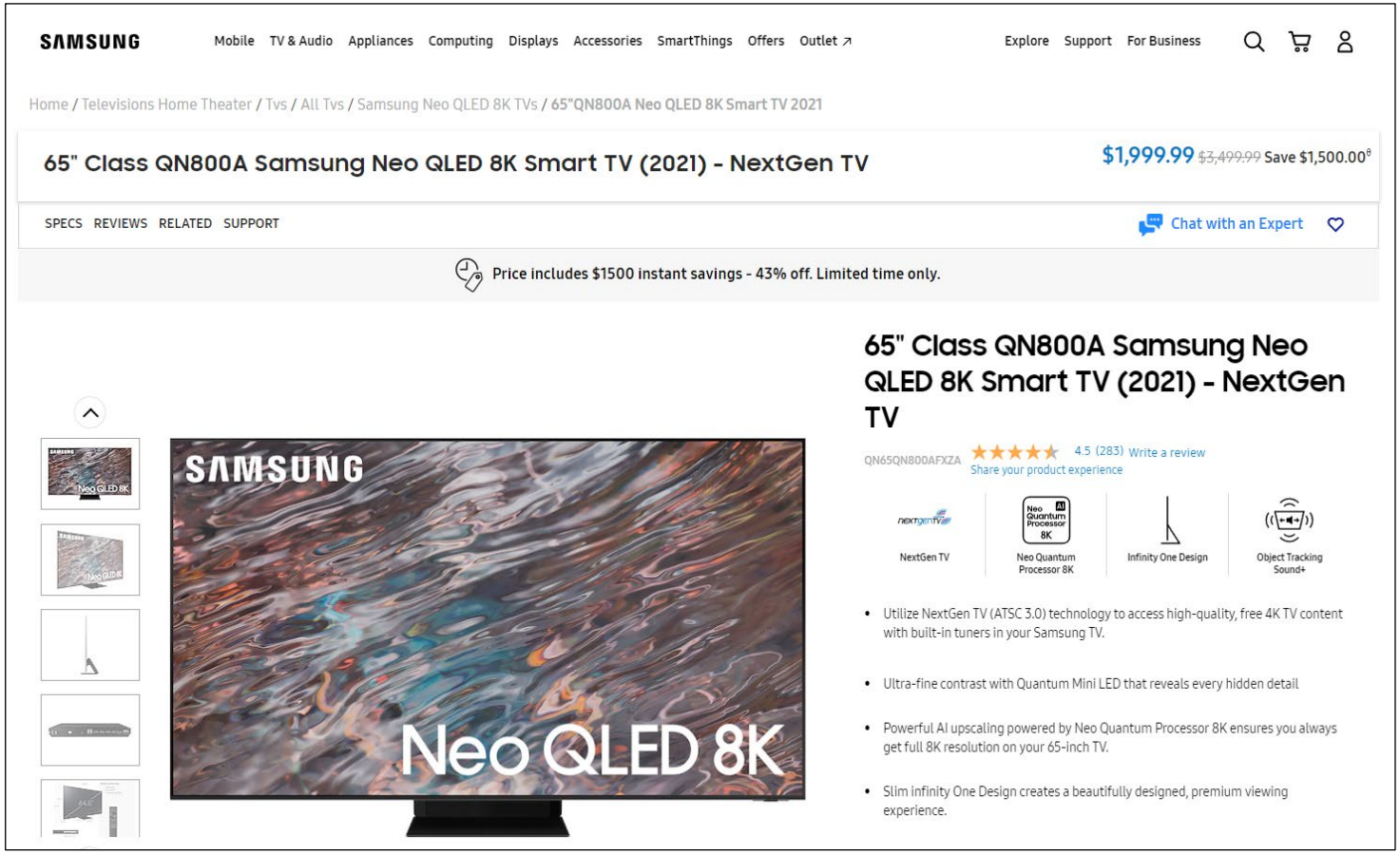
[8C] a decoding controlling program code to receive decoding control data for decoding control from the parameter data and control the selector to switch the predictive picture to be supplied to the combiner between the first and second predictive pictures according to the decoding control data.	The Samsung Accused Products comprise: a decoding controlling program code to receive decoding control data for decoding control from the parameter data and control the selector to switch the predictive picture to be supplied to the combiner between the first and second predictive pictures according to the decoding control data. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder uses read motion mode syntax as decoding controlling program code to receive decoding control data for decoding control from the parameter data (*e.g.*, parameters indicating motion mode) and control the selector to switch the predictive picture to be supplied to the combiner between the first (*i.e.*, first predictive picture using OBMC) and second predictive pictures (second predictive picture using translational motion compensation) according to the decoding control data. 5.11. Tile group OBU syntax

5.11.27. Read motion mode syntax

read_motion_mode(isCompound) {	Type
if (skip_mode) {	
motion_mode = SIMPLE	
return	
}	
if (!is_motion_mode_switchable) {	
motion_mode = SIMPLE	
return	
}	
if (Min(Block_Width[MiSize],	
Block_Height[MiSize]) < 8) {	
motion_mode = SIMPLE	
return	
}	
if (!force_integer_mv &&	
(YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV)) {	
if (GmType[RefFrame[0]] > TRANSLATION) {	
motion_mode = SIMPLE	
return	
}	
}	
if (isCompound || RefFrame[1] == INTRA_FRAME || !has_overlappable_candidates( )) {	
motion_mode = SIMPLE	
return	
}	
find_warp_samples()	
if (force_integer_mv || NumSamples == 0 ||	
!allow_warped_motion || is_scaled(RefFrame[0])) {	
use_obmc	S()
motion_mode = use_obmc ? OBMC : SIMPLE	
} else {	
motion_mode	S()

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10P] A moving-picture decoding method to be implemented in a moving-picture coding apparatus comprising the steps of:	Users of the Samsung Accused Products perform: A moving-picture decoding method implemented in a moving-picture coding apparatus. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A is a moving-picture decoding apparatus that implements moving picture decoding using the AV1 codec.



https://www.samsung.com/us/televisions-home-theater/tvs/samsung-neo-qled-8k/65-qn800a-neo-qled-8k-smart-tv-2021-qn65qn800afxza/

Compatible Samsung TVs

Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec.

TV Series	Models
Neo QLED 8K	QN900A, QN800A, QN700A
QLED 8K	Q950T, Q900T, Q800T, Q700T
Neo QLED 4K	QN95A, QN91A, QN90A, QN85A
QLED 4K	Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T
Crystal UHD	AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000
The Serif	LS01T
The Frame	LS03A, LS03T

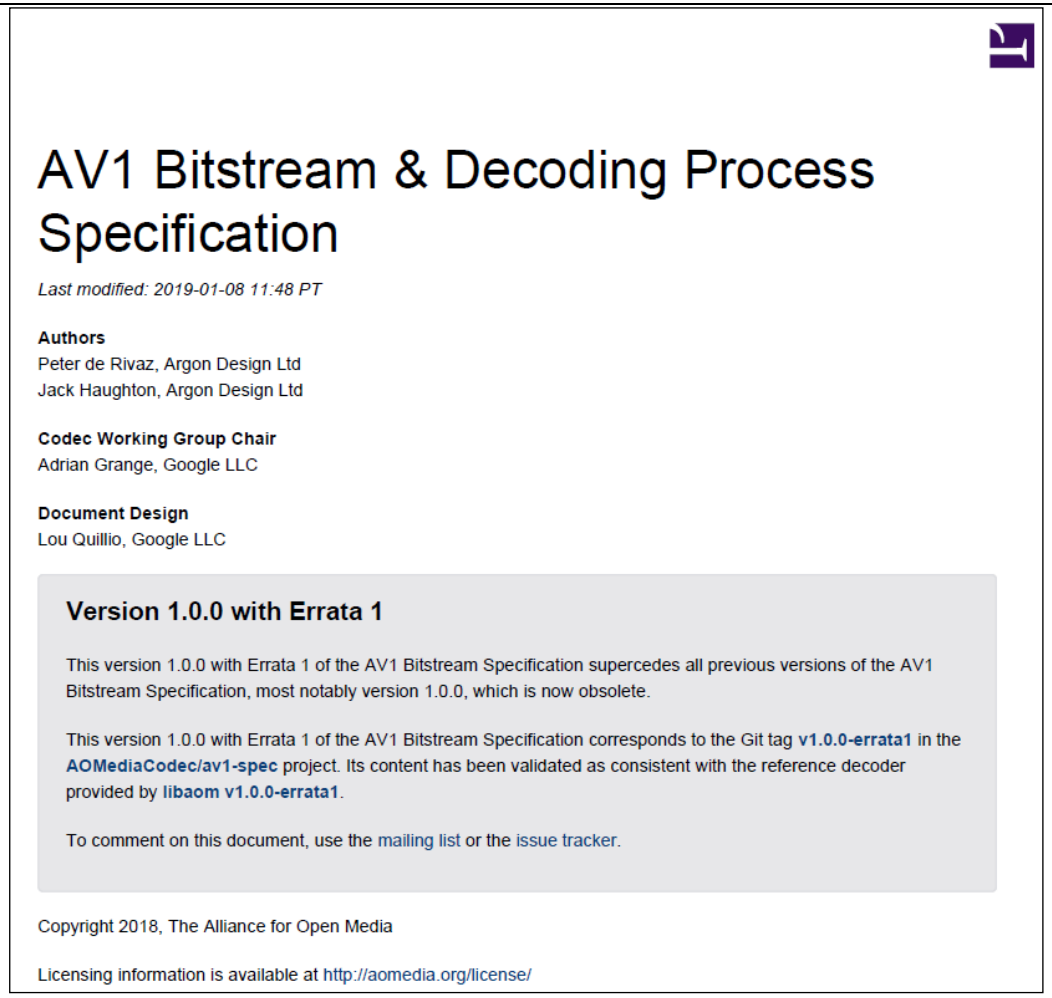
https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

The AV1 codec is used for video decoding, which is moving picture decoding.



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

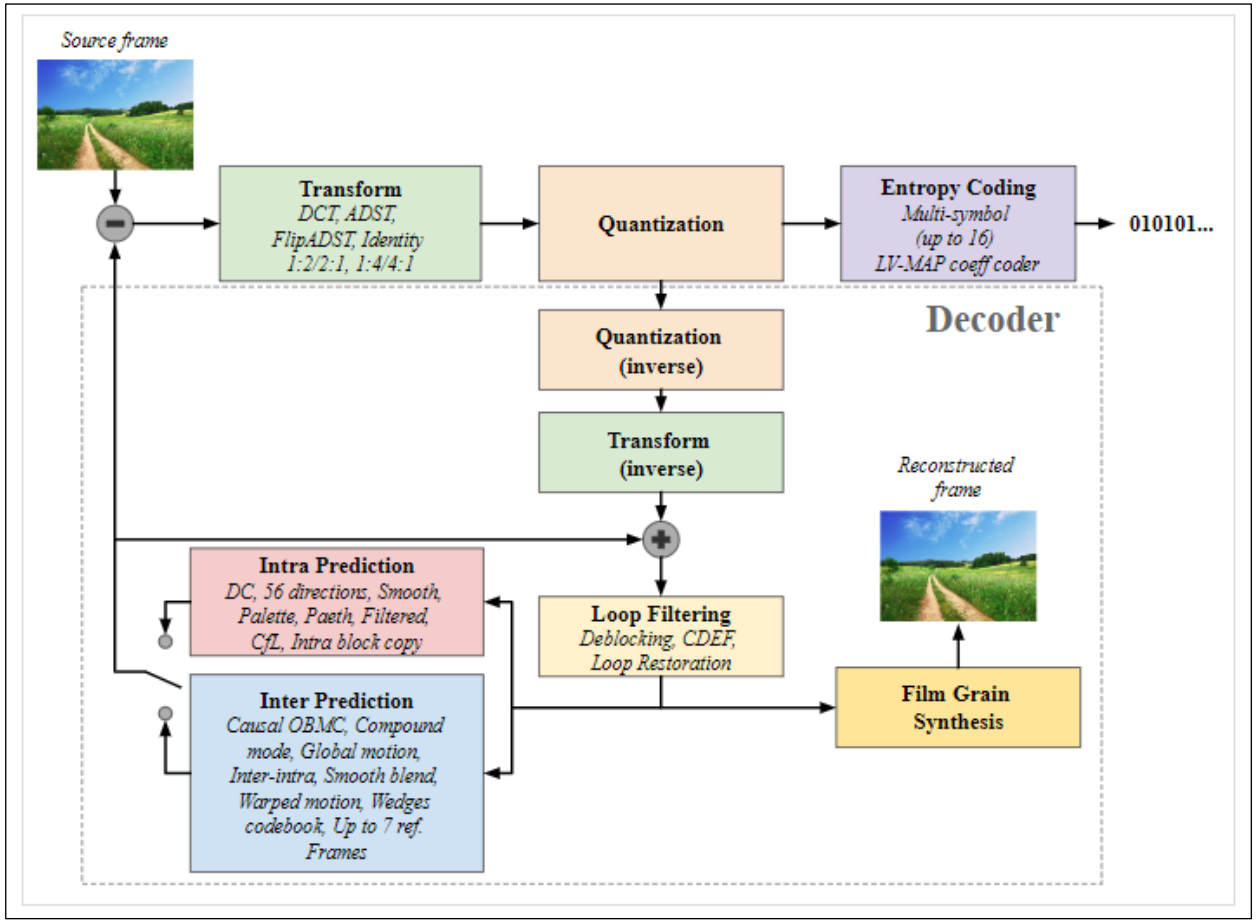
This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

	The AV1 decoder is a moving-picture coding apparatus that implements a moving-picture decoding method, as described below to reproduce a reconstructed frame.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10A1] demultiplexing coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding,	Users of the Samsung Accused Products perform: demultiplexing coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A demultiplexes coded data (.mp4, .mkv, or .webm files) from an input signal based on a specific syntax structure (open bitstream unit (OBU) syntax structure). **What is an AV1 file?** An AV1 file is a video encoded with AV1 compression, which is an open-source video codec developed by the Alliance for Open Media. AV1 videos are more commonly saved with the .MP4, .MKV, and .WEBM extension(s). https://fileinfo.com/extension/av1

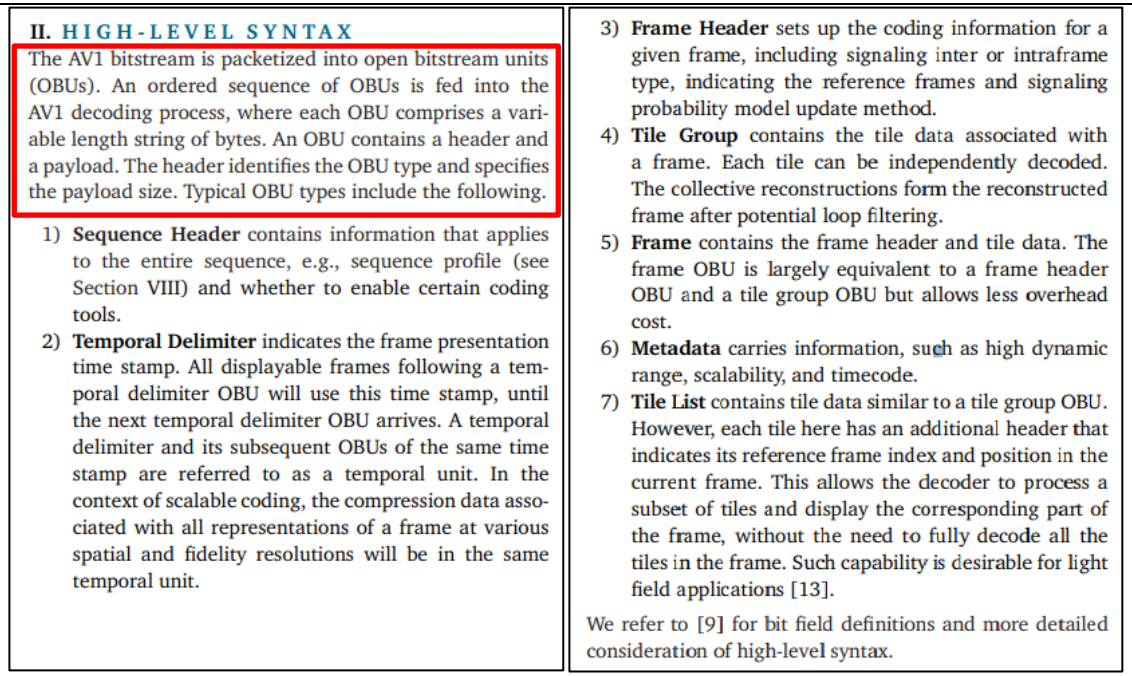
2022 TV Video Specifications

Samsung Smart TV provides rich capabilities for video playback, using the available audio and video decoders. This topic describes the media formats supported by Samsung Smart TV, and the limitations on their support.

22TV_PREMIUM1 (8K)

File Extension	Container	Video Codec	Resolution	Frame Rate (fps)	Bit Rate (Mbps)	Audio Codec
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS	H.264 BP/MP/HP	4096x2160	60	60	AC3 LPCM ADPCM(IMA, MS) AAC HE-AAC WMA DD+ MPEG(MP3) MPEG-H AC-4 OPUS G.711(A-Law, μ-Law)
			1920x1080	120		
		HEVC (H.265 - Main, Main10)	7680x4320	60	100	
			3840x2160	120		
		Motion JPEG		30	80	
		MPEG-4 SP/ASP	1920x1080	60	20	
		Window Media Video 9 (VC-1)				
		MPEG-2				
		MPEG-1				
		Microsoft MPEG-4 version 1/2/3				
		Window Media Video 7 (WMV1) / Video 8 (WMV2)		30		
		H.263 Sorenson				
		VP6				
		AV1	7680x4320	60	80	
				120		
*.webm	WebM	VP8	1920x1080	60	20	Vorbis
		VP9 (Profile 0/2)	3840x2160		80	
		AV1	7680x4320		60	
*.rmvb	RMVB	RV8/9/10 (RV30/40)	1920x1080		20	RealAudio 6

https://developer.samsung.com/smarttv/develop/specifications/media-specifications/2022-tv-video-specifications.html



II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

We refer to [9] for bit field definitions and more detailed consideration of high-level syntax.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 input signal to Samsung TVs is obtained by multiplexing a coded bitstream obtained by predictive coding (*i.e.*, predictive-transform coding), border motion-vector data (*i.e.*, interframe motion compensation), and post-quantization data (*i.e.*, reconstructed samples based on dequantized coefficients and inverse transforms)) obtained by quantization in the predictive coding.

	The AV1 codec supports input video signals in the 4:0:0 (monochrome), 4:2:0, 4:2:2, and 4:4:4 formats. The allowed pixel representations are 8, 10, and 12 bit. The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D unitary transform to further remove the spatial correlations, and the transform coefficients are quantized. Both the prediction syntax elements and the quantized transform coefficient indexes are then entropy coded using arithmetic coding. There are three optional in-loop postprocessing filter stages to enhance the quality of the reconstructed frame for reference by subsequent coded frames. A normative film grain synthesis unit is also available to improve the perceptual quality of the displayed frames. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 As shown below, the AV1 input signal is obtained by predictive coding (*i.e.*, predictive-transform coding). One example of predictive coding is through inter frame prediction:

V. CODING BLOCK OPERATIONS

C. Inter Frame Prediction

Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector.

Overlapped Block Motion Compensation The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

The variable isCompound is set equal to RefFrames[candRow][candCol][1] > INTRA_FRAME.

The prediction arrays are formed by the following ordered steps:

1. The rounding variables derivation process specified in section 7.11.3.2 is invoked with the variable isCompound as input.
2. If plane is equal to 0 and motion_mode is equal to LOCALWARP, the warp estimation process in section 7.11.3.8 is invoked.
3. If plane is equal to 0 and motion_mode is equal to LOCALWARP and LocalValid is equal to 1, the setup shear process specified in section 7.11.3.6 is invoked with LocalWarpParams as input, and the output warpValid is assigned to LocalValid (the other outputs are discarded).
4. The variable refList is set equal to 0.
5. The variable refFrame is set equal to RefFrames[candRow][candCol][refList].

6. If (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV) and GmType[refFrame] > TRANSLATION, the setup shear process specified in section 7.11.3.6 is invoked with gm_params[refFrame] as input, and the output warpValid is assigned to globalValid (the other outputs are discarded).

7. The variable useWarp (a value of 1 indicates local warping, 2 indicates global warping) is derived as follows:

    - If w < 8 or h < 8, useWarp is set equal to 0.
    - Otherwise, if force_integer_mv is equal to 1, useWarp is set equal to 0.
    - Otherwise, if motion_mode is equal to LOCALWARP and LocalValid is equal to 1, useWarp is set equal to 1.
    - Otherwise, if all of the following are true, useWarp is set equal to 2.
        - (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV).
        - GmType[refFrame] > TRANSLATION.
        - is_scaled(refFrame) is equal to 0.
        - globalValid is equal to 1.
    - Otherwise, useWarp is set equal to 0.

8. The motion vector array mv is set equal to Mvs[candRow][candCol][refList].

9. The variable refIdx specifying which reference frame is being used is set as follows:

    - If use_intrabc is equal to 0, refIdx is set equal to ref_frame_idx[refFrame - LAST_FRAME].
    - Otherwise (use_intrabc is equal to 1), refIdx is set equal to -1 and RefFrameWidth[-1] is set equal to FrameWidth, RefFrameHeight[-1] is set equal to FrameHeight, and RefUpscaledWidth[-1] is set equal to UpscaledWidth. (These values ensure that the motion vector scaling has no effect.)

10. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, x, y, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY.

11. If use_intrabc is equal to 1, RefFrameWidth[-1] is set equal to MiCols * MI_SIZE, RefFrameHeight[-1] is set equal to MiRows * MI_SIZE, and RefUpscaledWidth[-1] is set equal to MiCols * MI_SIZE. (These values are needed to avoid intrabc prediction being cropped to the frame boundaries.)

12. If useWarp is not equal to 0, the block warp process in section 7.11.3.5 is invoked with useWarp, plane, refList, x, y, i8, j8, w, h as inputs and the output is merged into the 2D array preds[refList] for i8 = 0..((h-1) >> 3) and for j8 = 0..((w-1) >> 3). (Each invocation fills in a block of output of size w by h at x offset j8 * 8 and y offset i8 * 8.)

13. If useWarp is equal to 0, the block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, w, h, candRow, candCol as inputs and the output is assigned to the 2D array preds[refList].

14. If isCompound is equal to 1, then the variable refList is set equal to 1 and steps 5 to 13 are repeated to form the prediction for the second reference.

An array named Mask is prepared as follows:

- If compound_type is equal to COMPOUND_WEDGE and plane is equal to 0, the wedge mask process in section 7.11.3.11 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_INTRA, the intra mode variant mask process in section 7.11.3.13 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_DIFFWTD and plane is equal to 0, the difference weight mask process in section 7.11.3.12 is invoked with preds, w, h as inputs.
- Otherwise, no mask array is needed.

If compound_type is equal to COMPOUND_DISTANCE, the distance weights process in section 7.11.3.15 is invoked with candRow and candCol as inputs.

The inter predicted samples are then derived as follows:

- If isCompound is equal to 0 and IsInterIntra is equal to 0, CurrFrame[plane][y + i][x + j] is set equal to Clip1(preds[0][i][j]) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_AVERAGE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(preds[0][i][j] + preds[1][i][j], 1 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_DISTANCE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(FwdWeight * preds[0][i][j] + BckWeight * preds[1][i][j], 4 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise, the mask blend process in section 7.11.3.14 is invoked with preds, plane, x, y, w, h as inputs.

If motion_mode is equal to OBMC, the overlapped motion compensation in section 7.11.3.9 is invoked with plane, w, h as inputs.

7.11.3.4. Block inter prediction process

The inputs to this process are:

- a variable plane,
- a variable refIdx specifying which reference frame is being used (or -1 for intra block copy),
- variables x and y giving the block location with in units of 1/1024 th of a sample,
- variables xStep and yStep giving the step size in units of 1/1024 th of a sample,
- variables w and h giving the width and height of the block in units of samples,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The output from this process is the 2D array named pred containing inter predicted samples.

A variable ref specifying the reference frame contents is set as follows:

- If refIdx is equal to -1, ref is set equal to CurrFrame.
- Otherwise (refIdx is greater than or equal to 0), ref is set equal to FrameStore[refIdx].

The variables subX and subY are set equal to the subsampling for the current plane as follows:

- If plane is equal to 0, subX is set equal to 0 and subY is set equal to 0.
- Otherwise, subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The variable lastX is set equal to ((RefUpscaledWidth[refIdx] + subX) >> subX) - 1.

The variable lastY is set equal to ((RefFrameHeight[refIdx] + subY) >> subY) - 1.

(lastX and lastY specify the coordinates of the bottom right sample of the reference plane.)

The variable intermediateHeight specifying the height required for the intermediate array is set equal to (((h - 1) * yStep + (1 << SCALE_SUBPEL_BITS) - 1) >> SCALE_SUBPEL_BITS) + 8.

The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. The fractional parts of the motion vectors determine the filtering process. If the fractional part is zero, then the filtering is equivalent to a straight sample copy.

The filtering is applied as follows:

- The array intermediate is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 1 ]
if ( w <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < intermediateHeight; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = x + xStep * c
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              ref[ plane ] [ Clip3( 0, lastY, (y >> 10) + r - 3 ) ]
                           [ Clip3( 0, lastX, (p >> 10) + t - 3 ) ]
        intermediate[ r ][ c ] = Round2(s, InterRound0)
    }
}
```

- The array pred is specified as follows:

```
interpFilter = InterpFilters[ candRow ][ candCol ][ 0 ]
if ( h <= 4 ) {
    if ( interpFilter == EIGHTTAP || interpFilter == EIGHTTAP_SHARP ) {
        interpFilter = 4
    } else if ( interpFilter == EIGHTTAP_SMOOTH ) {
        interpFilter = 5
    }
}
for ( r = 0; r < h; r++ ) {
    for ( c = 0; c < w; c++ ) {
        s = 0
        p = (y & 1023) + yStep * r
        for ( t = 0; t < 8; t++ )
            s += Subpel_Filters[ interpFilter ][ (p >> 6) & SUBPEL_MASK ][ t ] *
              intermediate[ (p >> 10) + t ][ c ]
        pred[ r ][ c ] = Round2(s, InterRound1)
    }
}
```

	https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257-263 As shown below, the AV1 input signal is obtained by border motion-vector data (*i.e.*, interframe motion compensation): V. CODING BLOCK OPERATIONS *C. Inter Frame Prediction* Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

	When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector: 1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs.

The function get_obmc_mask returns a blending mask as follows:

```
get_obmc_mask( length ) {
    if ( length == 2 ) {
        return Obmc_Mask_2
    } else if ( length == 4 ) {
        return Obmc_Mask_4
    } else if ( length == 8 ) {
        return Obmc_Mask_8
    } else if ( length == 16 ) {
        return Obmc_Mask_16
    } else {
        return Obmc_Mask_32
    }
}
```

The blending masks are defined as follows:

```
Obmc_Mask_2[2] = { 45, 64 }

Obmc_Mask_4[4] = { 39, 50, 59, 64 }

Obmc_Mask_8[8] = { 36, 42, 48, 53, 57, 61, 64, 64 }

Obmc_Mask_16[16] = { 34, 37, 40, 43, 46, 49, 52, 54,
                     56, 58, 60, 61, 64, 64, 64, 64 }

Obmc_Mask_32[32] = { 33, 35, 36, 38, 40, 41, 43, 44,
                     45, 47, 48, 50, 51, 52, 53, 55,
                     56, 57, 58, 59, 60, 60, 61, 62,
                     64, 64, 64, 64, 64, 64, 64, 64 }
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-278

As shown below, the AV1 input signal is obtained by post-quantization data (i.e., reconstructed samples based on dequantized coefficients and inverse transforms) obtained by quantization (i.e., using transform coefficients to discretize the signal) in the predictive coding:

Transform coefficient

A scalar value, considered to be in a frequency domain, contained in a transform block.

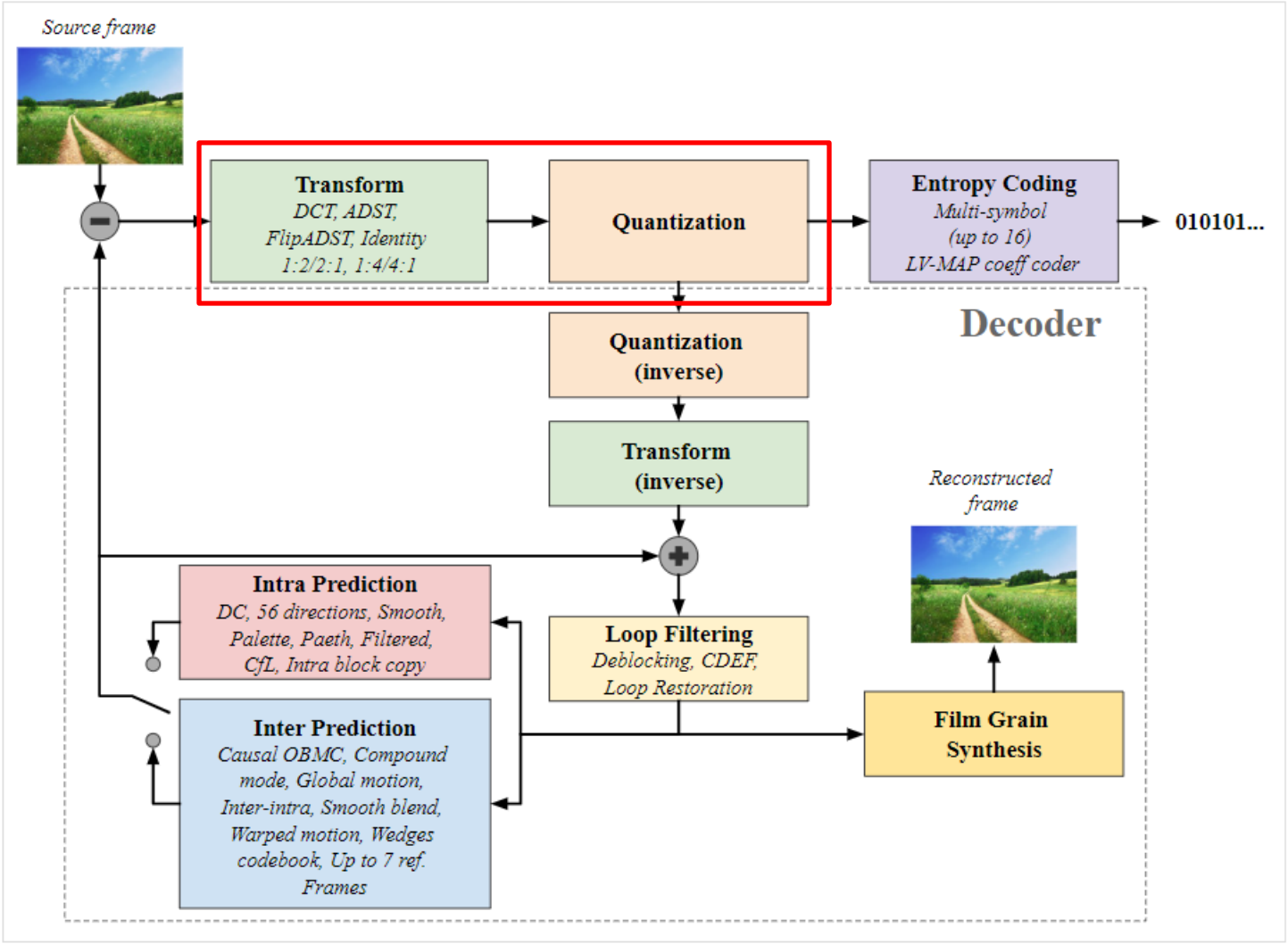
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 6

Quantized coefficient

A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4

	Dequantization The process in which transform coefficients are obtained by scaling the quantized coefficients. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10A2] the coded bitstream obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular	Users of the Samsung Accused Products perform: the coded bitstream obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the coded bitstream input to the Samsung 8K Neo QLED QN800A is obtained by producing and encoding a residual picture that is a residual signal (*i.e.*, prediction residue) between a picture to be coded (*i.e.*, original samples) that is an input moving-picture video signal to be subjected to coding and a predictive picture (*i.e.*, predicted samples) produced from a reference picture (*i.e.*, reference frame).

zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided,

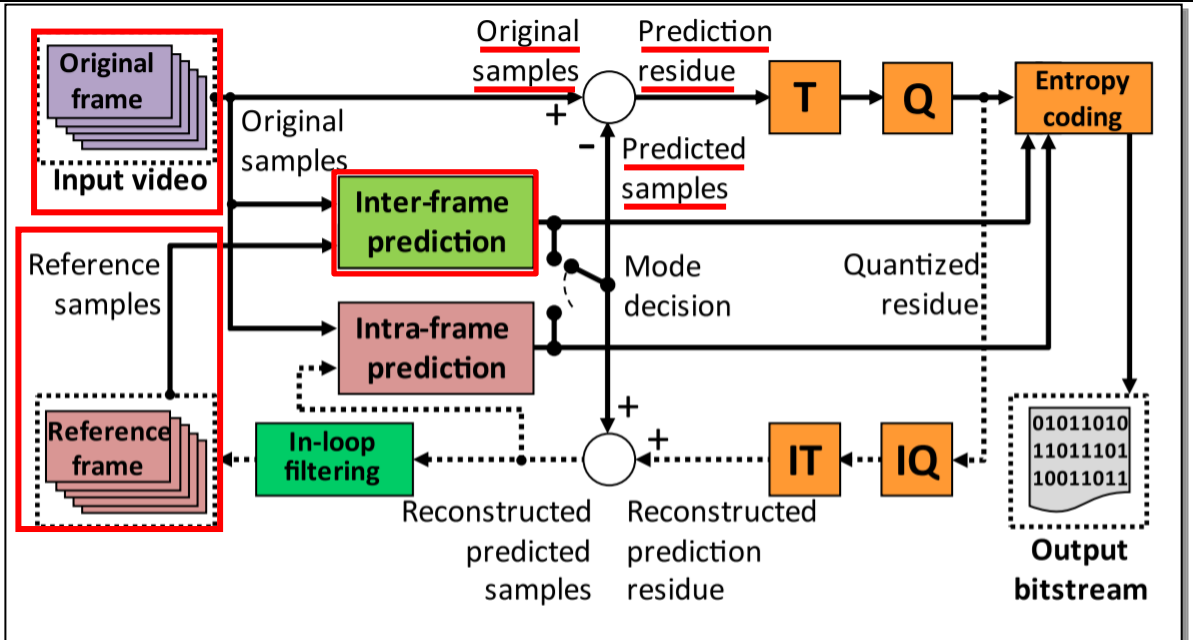


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

The reference picture (*i.e.*, reference frame) is a local decoded video signal (*i.e.*, previously decoded frame) for each of a plurality of rectangular zones (*i.e.*, block). The video area of the moving-picture video signal (*i.e.*, frame) is divided into such rectangular zones (*i.e.*, blocks).

Reference frame
A storage area for a previously decoded frame and associated information.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5

	Frame The representation of video signals in the spatial domain, composed of one luma sample matrix (Y) and two chroma sample matrices (U and V). https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 2 **Block** A square or rectangular region of samples. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 1 **Sample** The basic elements that compose the frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5 **Inter coding** Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 3 The plurality of rectangular zones (*i.e.*, several blocks of pixels) are composed of a specific number of pixels (*e.g.*, 128 x 128 pixels, 64 x 64 pixels, 4 x 4 pixels).

V. CODING BLOCK OPERATIONS

A. Coding Block Partitioning

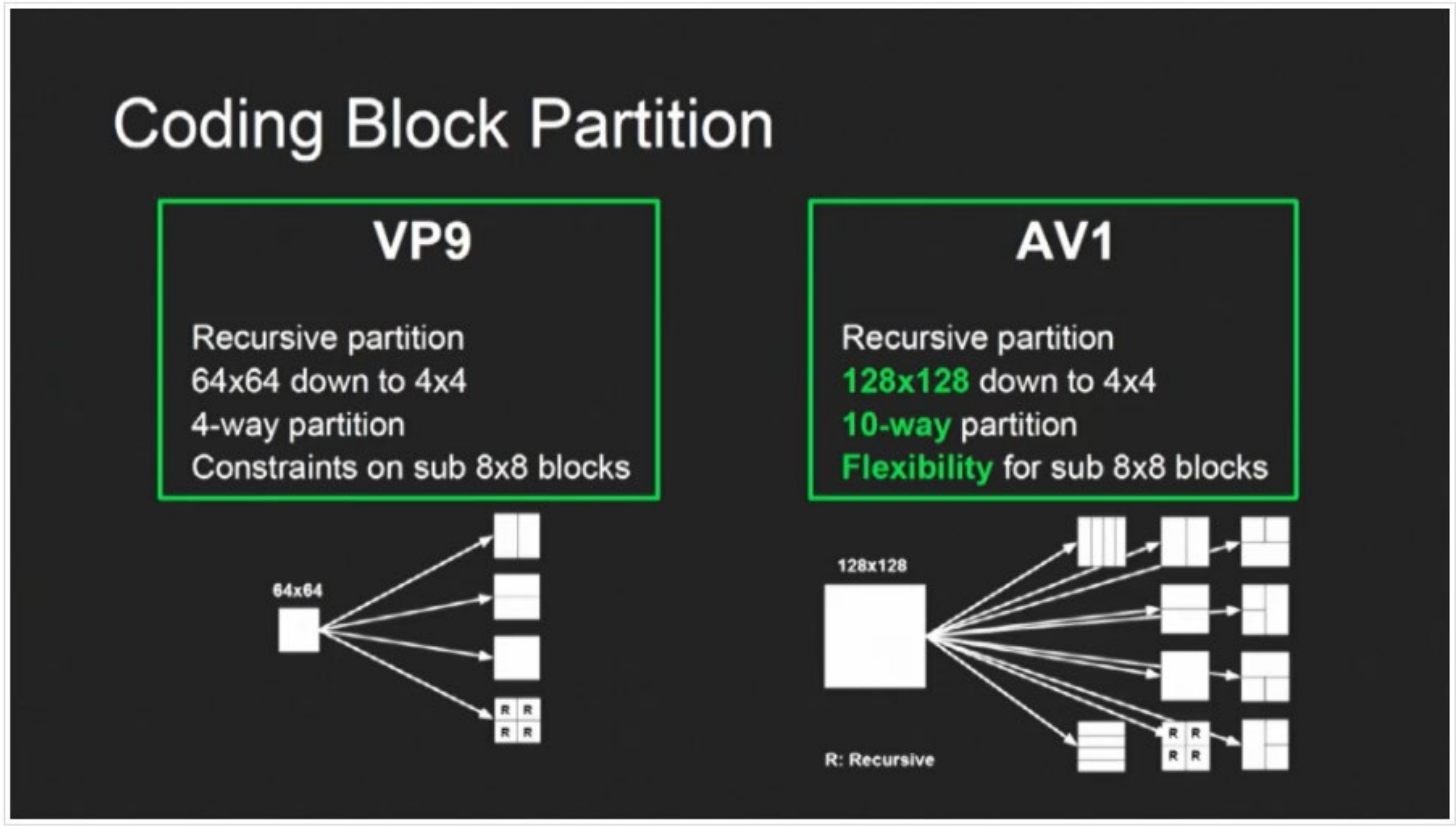
A superblock can be recursively partitioned into smaller block sizes for coding. AV1 inherits the recursive block partitioning design used in VP9. To reduce the overhead cost on prediction mode coding for video signals that are highly correlated, a situation typically seen in 4k videos, AV1 supports a maximum coding block size of 128×128 luma samples. The allowed partition options at each block level include ten possibilities, as shown in Fig. 2. To improve the prediction quality for complex videos, the minimum coding block size is extended to 4×4 luma samples. While such extensions provide more coding flexibility, they have implications for hardware decoders. Certain block size-dependent constraints are specifically designed to circumvent such complications.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D uni-

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	A. BLOCK PARTITIONING Before the prediction process begins, a frame is divided into several blocks of pixels. Each video encoder defines a variable range of block sizes it can use. In AV1, a video frame is partitioned in superblocks (SBs) of size 128×128 or 64×64 pixels. To deliver a locally optimal prediction for each SB, the encoder can further divide each SB using a 10-way partition tree structure, as illustrates Fig. 2. In the figure, filled partitions are final partition modes, but all four partitions of the unfilled partition (SPLIT) can be recursively divided based on the same 10-way tree structure, down to 4×4, which is the smallest supported block size [2]. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

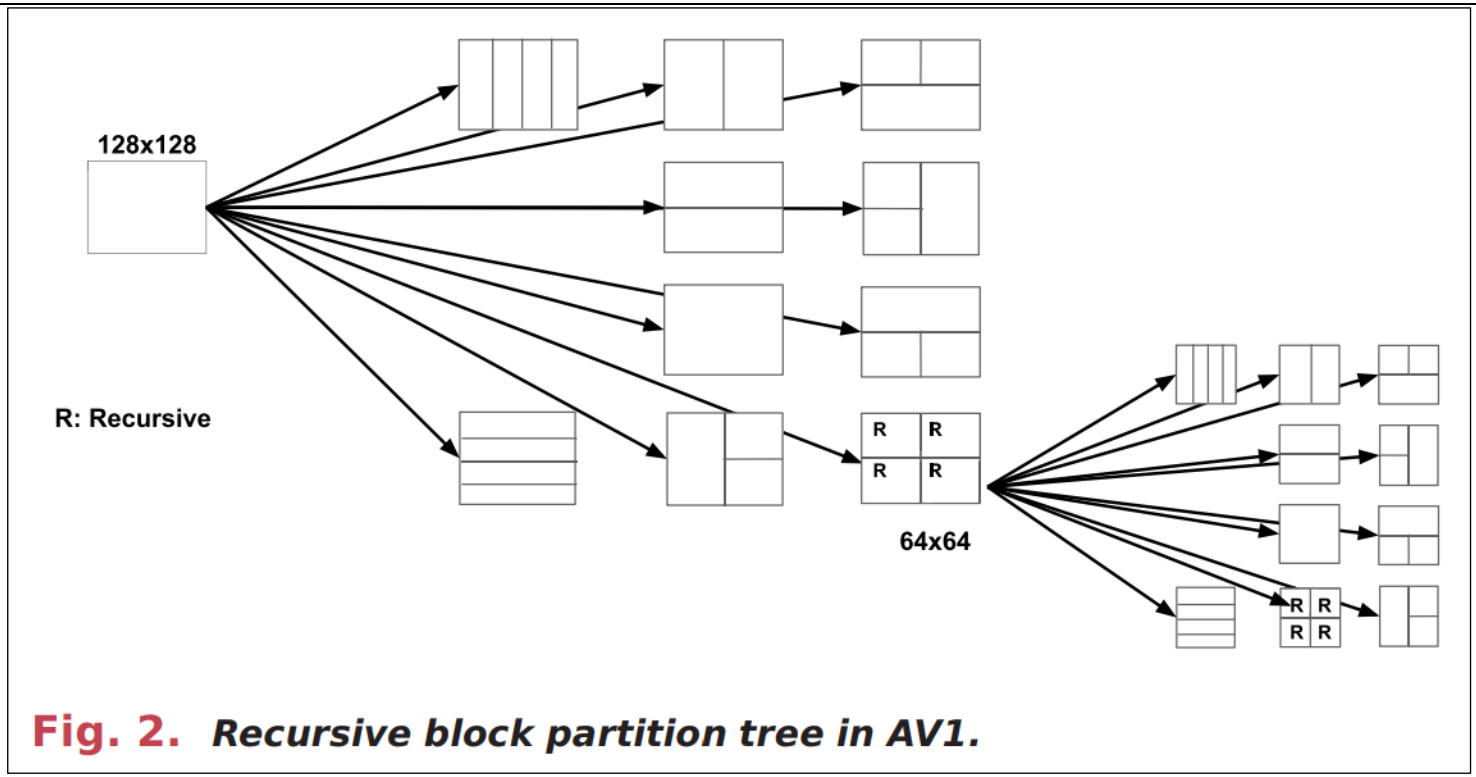
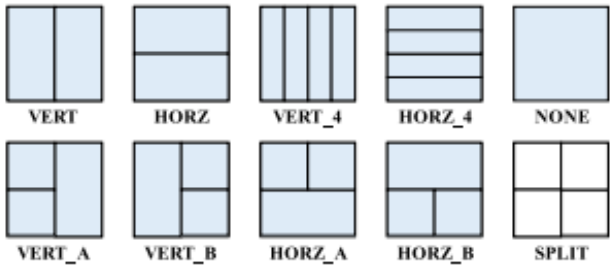


Fig. 2. ***Recursive block partition tree in AV1.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	 FIGURE 2. AV1 10-way block partition tree structure. https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8= Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10A3] obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular	Users of the Samsung Accused Products perform: obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A, through the use of the AV1's block-matching algorithms, such as overlapped block motion compensation (OBMC), obtains the boundary conditions of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones (*i.e.*, "immediate spatial neighbors' motion information to improve the prediction quality for pixels near [a

zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found,	block's] top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides"). After the boundary condition of the block is obtained, the OBMC finds a boundary condition in the reference frame that matches the boundary condition in the current frame to identify the "four reference blocks that have the same reference frame as the current block" and ensure that the boundary condition is met (*i.e.*, "width is the same as the reference block's width, and its height is half of the current block's height"). *e) Overlapped block motion compensation:* The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

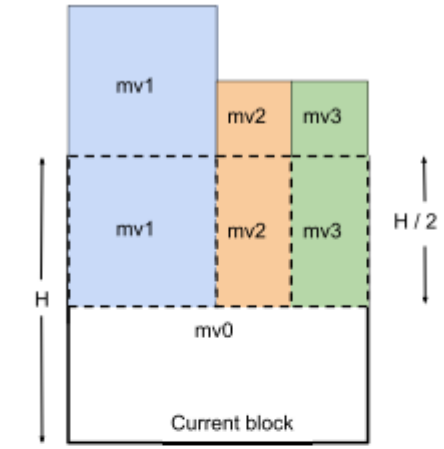


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

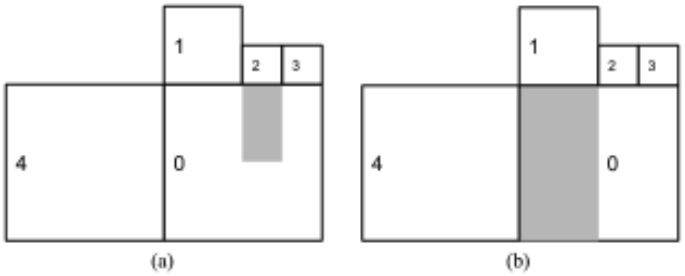


Fig. 1. Overlapping regions

https://ieeexplore.ieee.org/document/8296419

Additionally, for example, the Samsung 8K Neo QLED QN800A, through the use of the AV1 decoder generates border motion-vector data to be coded to the border of the reference picture (*i.e.*, reference frame) because the motion vector is defined as a "two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame." Such a motion vector is generated at each location on an 8x8 luma sample grid. The purpose of such generated motion vectors is to further generate a motion compensation block that extends from the top boundary to toward the center of the current block.

Motion vector

A two-dimensional vector used for inter prediction which refers the current frame to the reference frame, the value of which provides the coordinate offsets from a location in the current frame to a location in the reference frame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

7.9. Motion field estimation process

7.9.1. General

This process is triggered by a call to motion_field_estimation while reading the uncompressed header.

A linear projection model is employed to create a motion field estimation that is able to capture high velocity temporal motion trajectories.

The motion field is estimated based on the saved motion vectors from the reference frames and the relative frame distances.

As the frame distances depend on the frame being referenced, a separate motion field is estimated for each reference frame used by the current frame.

A motion vector (for each reference frame type) is prepared at each location on an 8x8 luma sample grid.

The variable w8 (representing the width of the motion field in units of 8x8 luma samples) is set equal to MiCols >> 1.

The variable h8 (representing the height of the motion field in units of 8x8 luma samples) is set equal to MiRows >> 1.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 212

e) Overlapped block motion compensation: The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides.

It first scans through the immediate neighbors above and finds up to four reference blocks that have the same reference frame as the current block. An example is shown in Fig. 14(a), where the blocks are marked according to their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

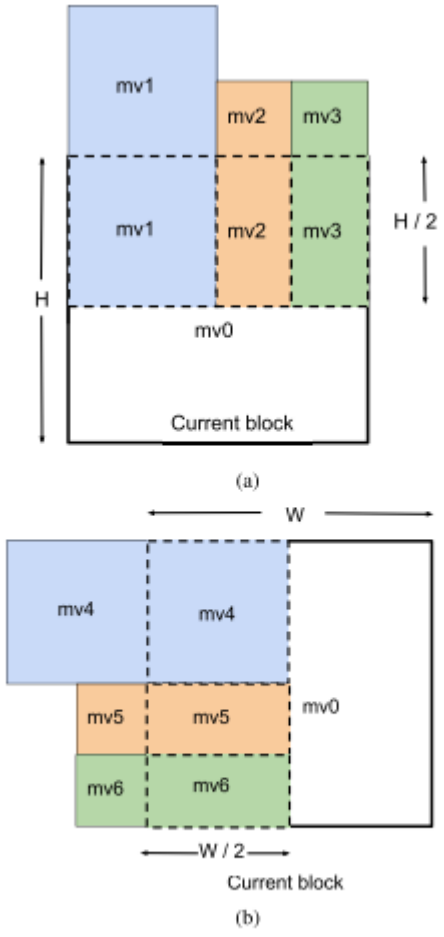


Fig. 14. ***Overlapped block motion compensation using (a) top and (b) left neighboring blocks' motion information, respectively.***

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

https://ieeexplore.ieee.org/document/8296419

	Motion vector coding accounts for a sizable portion of the overall bit rate. Modern video codecs typically adopt predictive coding for motion vectors and code the difference using entropy coding [30], [31]. The prediction accuracy has a large impact on the coding efficiency. AV1 employs a dynamic motion vector referencing scheme that obtains candidate motion vectors from the spatial and temporal neighbors and ranks them for efficient entropy coding. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10A4] defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border	Users of the Samsung Accused Products perform: defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec, which is described above with respect to [10A3] and is incorporated here by reference.

motion-vector data, and	Specifically, a boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (i.e., immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (i.e., motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10A5] generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies	Users of the Samsung Accused Products perform: generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing the predictive picture. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Poisson's Equation, thus producing the predictive picture;	For example, the Samsung 8K Neo QLED QN800A practices overlapped block motion compensation using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (i.e., prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (i.e., that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

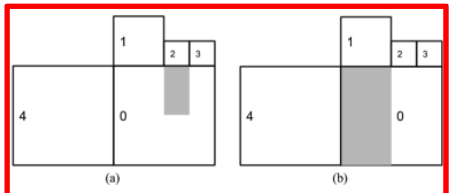


Fig. 1. Overlapping regions

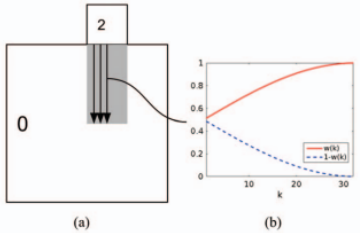


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

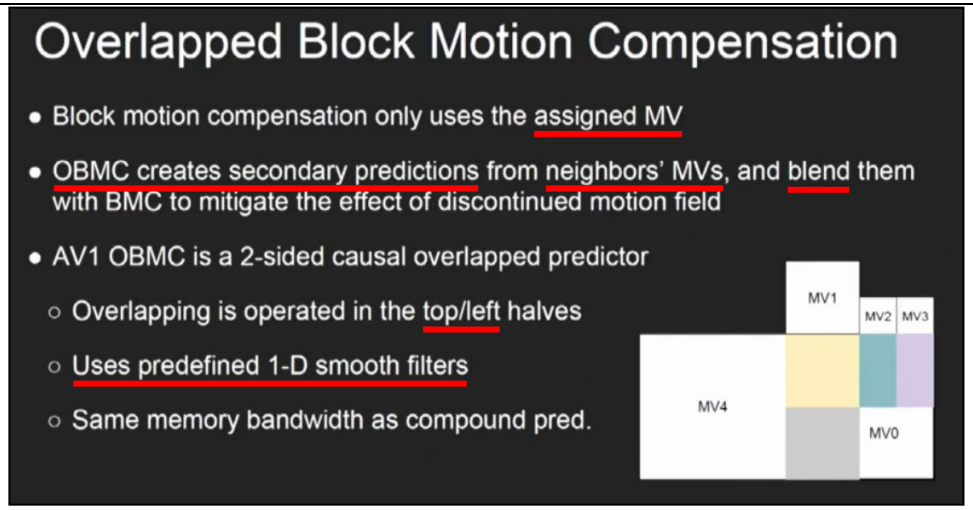
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

385

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

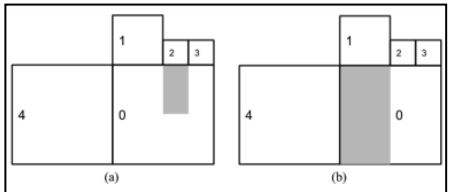


Fig. 1. Overlapping regions

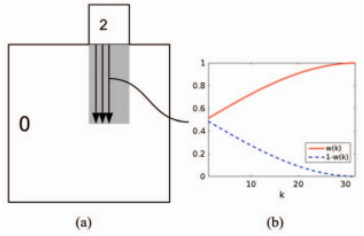


Fig. 2. Weighting based on 1-D raised cosine window

386

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

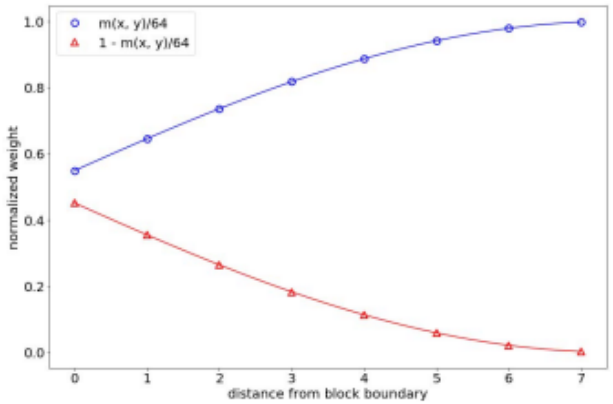


Fig. 15. ***Normalized weights for OBMC with H – 16 or W – 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10B] performing entropy decoding to the data thus demultiplexed to generate, at least, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure;	Users of the Samsung Accused Products perform: performing entropy decoding to the data thus demultiplexed to generate, at least, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A performs entropy decoding to the demultiplexed data required for constructing a specific syntax structure. Entropy decoding in AV1 generates, at least, the post-quantization data (*i.e.*, quantized transform coefficients), the border motion vector data (*i.e.*, at least the "the motion vector difference [that] will be entropy coded"), and parameter data (*i.e.*, quantization parameters and other "lateral data") for constructing a specific syntax structure (*i.e.*, open bitstream units). Entropy decoding is performed in order to construct a specific syntax structure (*i.e.*, "to compress the syntax elements"). In other words, the "output of this [entropy decoding] process is the variable symbol, containing a decoded syntax element."

employed for transform blocks at dimension 32×32 and above.

F. Quantization

The transform coefficients are quantized, and the quantization indexes are entropy coded. The quantization parameter (QP) in AV1 ranges between 0 and 255.

1) Quantization Step Size: At a given QP, the quantization step size for the dc coefficient is smaller than that for the ac coefficient. The mapping from QP to quantization step size for both dc and ac coefficients is drawn in Fig. 25. The lossless coding mode is achieved when QP is 0. By default, all the ac coefficients will use the same quantization step size. Since the human visual system tends to have different tolerance to distortions at various frequencies, AV1 also supports 15 sets of predefined quantization weighting matrices, where the quantization step size for each individual frequency component is further scaled differently. Each frame can optionally select a quantization weighting matrix set for luma and chroma planes, respectively.

2) Quantization Parameter Modulation: AV1 assigns a base QP for a coded frame, denoted by QP_{base}. The QP values for the dc and ac coefficients in both luma and chroma components are shown in Table 1. $\Delta QP_{p,b}$ are additional offset values transmitted in the frame header,

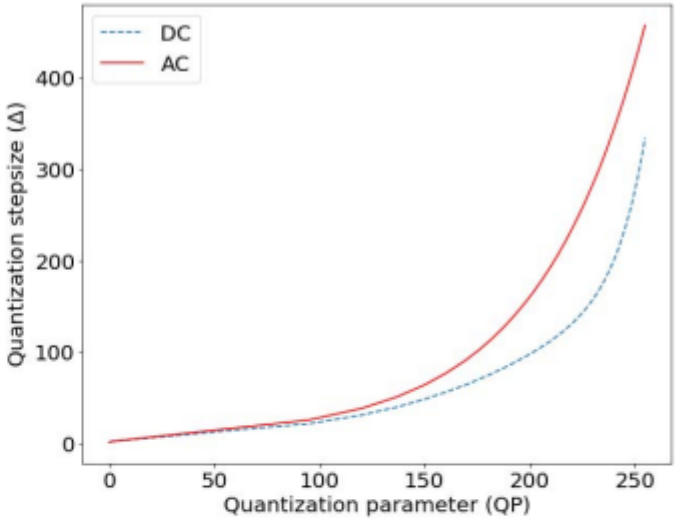


Fig. 25. ***QP and quantization step size maps for dc and ac coefficients.***

where $p \in \{Y, U, V\}$ denotes the plane and $b \in \{dc, ac\}$ denotes the dc or the ac transform coefficients.

Recognizing that the coding blocks within a frame may have different rate-distortion tradeoffs, AV1 further allows QP offset at both superblock and coding block levels. The resolution of QP offset at superblock level is signaled by the frame header. The available options are 1, 2, 4, and 8. The coding block-level QP offset can be achieved through

Table 1 Frame-Level QP Values (QP_{frame}) for Y/U/V Planes

	AC	DC
Y	QP_{base}	$QP_{base} + \Delta QP_{Y,DC}$
U	$QP_{base} + \Delta QP_{U,AC}$	$QP_{base} + \Delta QP_{U,DC}$
V	$QP_{base} + \Delta QP_{V,AC}$	$QP_{base} + \Delta QP_{V,DC}$

segmentation. AV1 allows a frame to classify its coding blocks into up to eight segments; each has its own QP offset decided by the frame header. The segment index associated with each coding block is sent through the bitstream to the decoder.

Therefore, the effective QP for ac coefficients in a coding block, QP_{cb}, is given by

$$QP_{cb} = \text{clip}(QP_{\text{frame}} + \Delta QP_{sb} + \Delta QP_{\text{seg}}, 1, 255) \quad (27)$$

where ΔQP_{sb} and ΔQP_{seg} are the QP offsets from the superblock and the segment, respectively. The clip function ensures it stays within a valid range. The QP is not allowed to change from a nonzero value to zero since zero is reserved for lossless coding.

The decoder rebuilds the quantized samples using a uniform quantizer. Given the quantization step size Δ and the quantization index k, the reconstructed sample is $k\Delta$.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

The first four motion vectors in this ranked list will be used as candidate motion vector predictors. The encoder will pick the one that is closest to the desired motion vector and send its index to the decoder. It is not uncommon for coding blocks to have fewer than four candidate motion vectors, due to either the high flexibility in the reference frame selection, or a highly consistent motion activity in the local region. In such a context, the candidate motion vector list will be shorter than 4, which allows the codec to save bits spent on identifying the selected index. The dynamic candidate motion vector list is in contrast to the design in VP9, where one always constructs two candidate motion vectors. If not enough candidates are found, the VP9 codec will fill the list with zero vectors. AV1 also supports a special intermode that makes the inter-predictor use the frame-level affine model, as discussed in Section V-C2.

The motion vector difference will be entropy coded. Since a significant portion of the coding blocks will find a zero motion vector difference, the probability model is designed to account for such bias. AV1 allows a coding block to use 1 bit to indicate whether to directly use the selected motion vector predictor as its final motion vector or to additionally code the difference. The probability model for this entropy-coded bit is conditioned on two factors: whether its spatial neighbors have a nonzero motion vector difference and whether a sufficient number of motion vector predictors are found. For compound modes, where two motion vectors need to be specified, this extends to four cases that cover where either block, both, or neither one have a zero difference motion vector. The nonzero difference motion vector coding is consistent in all cases.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

VI. ENTROPY CODING SYSTEM

AV1 employs an M-ary symbol arithmetic coding method that was originally developed for the Daala video codec [39] to compress the syntax elements, where integer $M \in [2, 14]$. The probability model is updated per symbol coding.

C. Level Map Transform Coefficient Coding System

The transform coefficient entropy coding system is an intricate and performance-critical component in video codecs. We discuss its design in AV1 that decomposes it into a series of symbol codings.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

E. ENTROPY CODING

The entropy coding processes the symbols (quantized coefficients and lateral data) to reduce their statistical redundancy by applying lossless algorithms.

AV1 uses a symbol-to-symbol adaptive multi-symbol arithmetic coder with the probability being updated every new symbol. Each syntax element in AV1 is a member of an alphabet of N elements, and a context consists of a set of N probabilities together with a small count to facilitate fast early adaptation [2].

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

Quantization parameter
A variable used for scaling the quantized coefficients in the decoding process.

Quantized coefficient
A transform coefficient before dequantization.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

D. TRANSFORMS AND QUANTIZATION

The prediction error, or the residues, between the intra and inter prediction and the original blocks are processed by the transform module (T module, in Fig. 1), which converts the values from the spatial domain to the frequency domain. Then, the quantization step (Q module in Fig. 1) is applied to the transformed coefficients to attenuate or eliminate values associated with spectral components that are not perceptually relevant for the human visual system.

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

8.2. Parsing process for symbol decoder

8.2.1. General

The entropy decoder is referred to as the “Symbol decoder” and the functions init_symbol(sz), exit_symbol( ), read_symbol(cdf), and read_bool( ) are used in this Specification to indicate the entropy decoding operation.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

	8.2.2. Initialization process for symbol decoder The input to this process is a variable sz specifying the number of bytes to be read by the Symbol decoder. This process is invoked when the function init_symbol(sz) is called from the syntax structure. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 353

8.2.4. Exit process for symbol decoder

This process is invoked when the function exit_symbol( ) is called from the syntax structure.

It is a requirement of bitstream conformance that SymbolMaxBits is greater than or equal to -14 whenever this process is invoked.

The variable trailingBitPosition is set equal to get_position() - Min(15, SymbolMaxBits+15).

The bitstream position indicator is advanced by Max(0,SymbolMaxBits). (This skips over any trailing bits that have not already been read during symbol decode.)

The variable paddingEndPosition is set equal to get_position().

> **Note:** paddingEndPosition will always be a multiple of 8 indicating that the bit position is byte aligned.

It is a requirement of bitstream conformance that the bit at position trailingBitPosition is equal to 1.

It is a requirement of bitstream conformance that the bit at position x is equal to 0 for values of x strictly between trailingBitPosition and paddingEndPosition.

> **Note:** This exit process consumes the OBU trailing bits for a tile group.

If disable_frame_end_update_cdf is equal to 0 and TileNum is equal to context_update_tile_id, a copy is made of the final CDF values for each of the CDF arrays mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs. The name of the destination for the copy is the name of the CDF array prefixed with “Saved”. The name of the source for the copy is the name of the CDF array prefixed with “Tile”. For example, an array SavedYModeCdf will be created with values equal to TileYModeCdf.

	https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 358-359

8.2.6. Symbol decoding process

The input to this process is an array cdf of length N + 1 which specifies the cumulative distribution for a symbol with N possible values.

The output of this process is the variable symbol, containing a decoded syntax element. The process also modifies the input array cdf to adapt the probabilities to the content of the stream.

This process is invoked when the function read_symbol(cdf) is called.

> **Note:** When this process is invoked, N will be greater than 1 and cdf[N-1] will be equal to 1 << 15.

The variables cur, prev, and symbol are calculated as follows:

```
cur = SymbolRange
symbol = -1
do {
  symbol++
  prev = cur
  f = ( 1 << 15 ) - cdf[ symbol ]
  cur = ((SymbolRange >> 8) * (f >> EC_PROB_SHIFT)) >> (7 - EC_PROB_SHIFT)
  cur += EC_MIN_PROB * (N - symbol - 1)
} while ( SymbolValue < cur )
```

Note: Implementations may prefer to store the inverse cdf to move the subtraction out of this loop.

The variable SymbolRange is set to prev - cur.

The variable SymbolValue is set equal to SymbolValue - cur.

The range and value are renormalized by the following ordered steps:

1. The variable bits is set to 15 – FloorLog2(SymbolRange). This represents the number of new bits to be added to SymbolValue.
2. The variable SymbolRange is set to SymbolRange << bits.
3. The variable numBits is set equal to Min(bits, Max(0, SymbolMaxBits)). This represents the number of new bits to read from the bitstream.
4. The variable newData is read using the f(numBits) parsing process.
5. The variable paddedData is set equal to newData << (bits - numBits).
6. The variable SymbolValue is set to paddedData ^ (((SymbolValue + 1) << bits) - 1).
7. The variable SymbolMaxBits is set to SymbolMaxBits - bits.

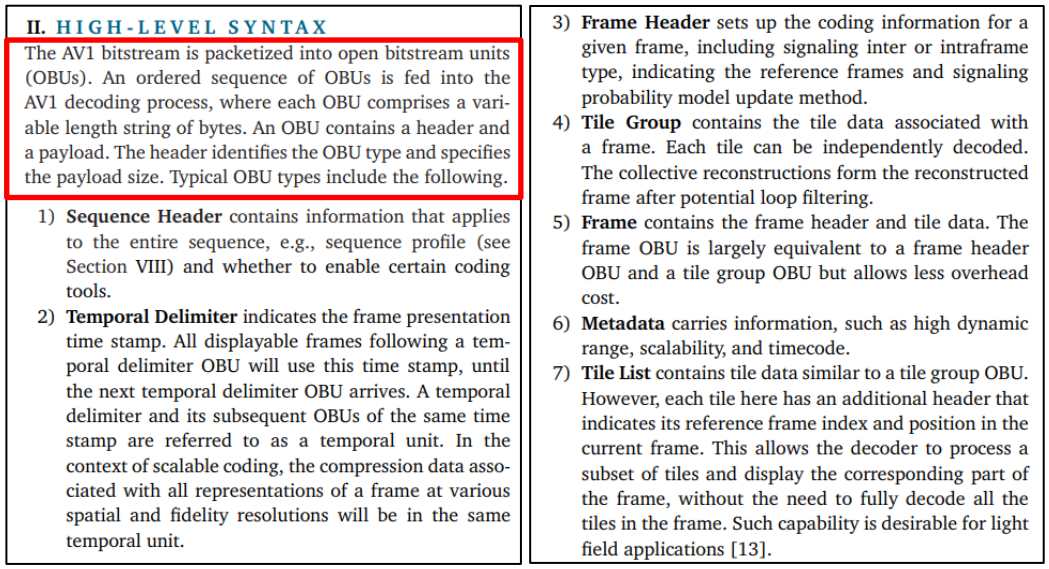
Note: bits may be equal to 0, in which case these ordered steps have no effect.

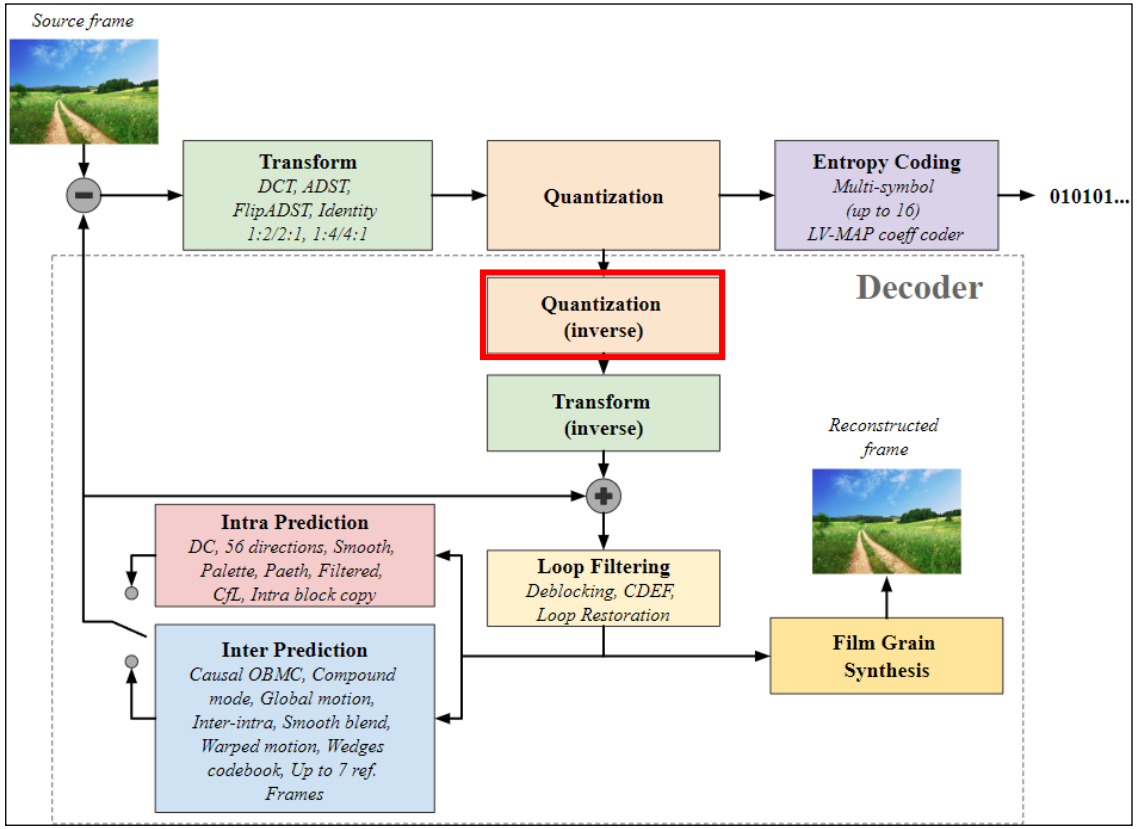
If disable_cdf_update is equal to 0, the cumulative distribution is updated as follows:

```
rate = 3 + ( cdf[ N ] > 15 ) + ( cdf[ N ] > 31 ) + Min( FloorLog2( N ), 2 )
tmp = 0
for ( i = 0; i < N - 1; i++ ) {
  tmp = ( i == symbol ) ? ( 1 << 15 ) : tmp
  if ( tmp < cdf[ i ] ) {
    cdf[ i ] -= ( ( cdf[ i ] - tmp ) >> rate )
  } else {
    cdf[ i ] += ( ( tmp - cdf[ i ] ) >> rate )
  }
}
cdf[ N ] += ( cdf[ N ] < 32 )
```

Note: The last entry of the cdf array is used to keep a count of the number of times the symbol has been decoded (up to a maximum of 32). This allows the cdf adaption rate to depend on the number of times the symbol has been decoded.

The return value from the function is given by symbol.

	https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 359-361  **II. HIGH-LEVEL SYNTAX** The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following. 1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools. 2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit. 3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method. 4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering. 5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost. 6) **Metadata** carries information, such as high dynamic range, scalability, and timecode. 7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10C] performing inverse-quantization to the post-quantization data to generate	Users of the Samsung Accused Products perform: inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

post-quantization orthogonal transform coefficients data;	For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec to perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ **Dequantization** The process in which transform coefficients are obtained by scaling the quantized coefficients. https://aomediacodec.github.io/av1-spec/av1-spec.pdf

401

II. AV1 AND VVC FEATURES

Both the AV1 and VVC encoders follow a hybrid block-based video coding scheme, which is based on the following signal and data processing operations, as presented in Fig. 1: (i) inter and intra-frame prediction, (ii) transform (T), (iii) quantization (Q), and (iv) entropy coding [11]. A reconstruction loop with inverse quantization (IQ) and inverse transform (IT) is also included to guarantee that the encoder and decoder will use the same reference data [11]. Finally,

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12. Reconstruction and dequantization

7.12.1. General

This section details the process of reconstructing a block of coefficients using dequantization and inverse transforms.

7.12.2. Dequantization functions

This section defines the functions get_dc_quant and get_ac_quant that are needed by the dequantization process.

The quantization parameters are derived from lookup tables.

	The function get_dc_quant(plane) returns the quantizer value for the dc coefficient for a particular plane and is derived as follows: • If plane is equal to 0, return dc_q(get_qindex(0, segment_id) + DeltaQYDc). • Otherwise if plane is equal to 1, return dc_q(get_qindex(0, segment_id) + DeltaQUDc). • Otherwise (plane is equal to 2), return dc_q(get_qindex(0, segment_id) + DeltaQVDc). The function get_ac_quant(plane) returns the quantizer value for the ac coefficient for a particular plane and is derived as follows: • If plane is equal to 0, return ac_q(get_qindex(0, segment_id)). • Otherwise if plane is equal to 1, return ac_q(get_qindex(0, segment_id) + DeltaQUAc). • Otherwise (plane is equal to 2), return ac_q(get_qindex(0, segment_id) + DeltaQVAc). https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-294 7.13. Inverse transform process 7.13.1. General This section details the inverse transforms used during the reconstruction processes detailed in section 7.12. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 296 The generated post-quantization transform coefficients data are orthogonal because at least some of the inverse transform processes are orthogonal (*e.g.*, Inverse Discrete Cosine Transform (DCT) process).

7.13.2.2. Inverse DCT array permutation process

This process performs an in-place permutation of the array T of length 2^n for $2 \le n \le 6$ which is required before execution of the inverse DCT process.

The input to this process is a variable n that specifies the base 2 logarithm of the length of the input array.

A temporary array named copyT is set equal to T.

T[i] is set equal to copyT[brev(n, i)] for i = 0..((1 << n) - 1).

7.13.2.3. Inverse DCT process

This process performs an in-place inverse discrete cosine transform of the permuted array T which is of length 2^n for $2 \le n \le 6$.

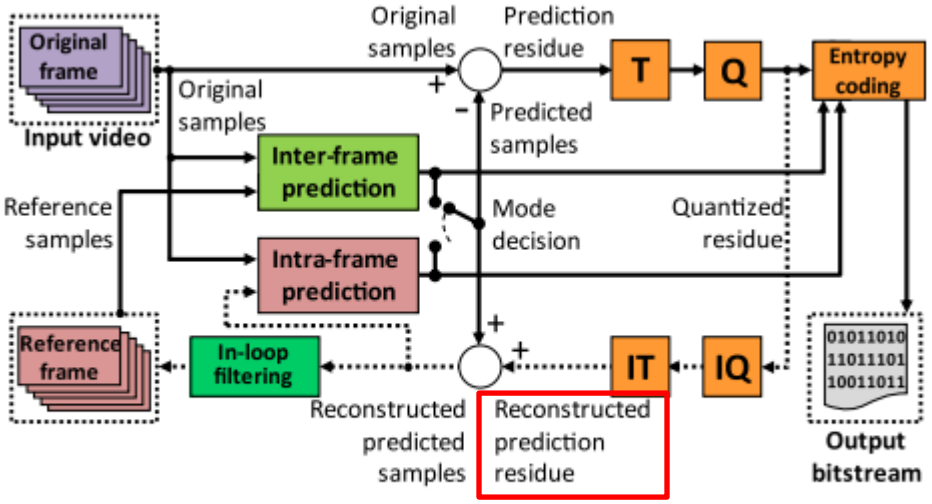
The inputs to this process are:

- a variable n that specifies the base 2 logarithm of the length of the input array,
- a variable r that specifies the intermediate clamping range.

The following ordered steps apply:

1. Invoke the inverse DCT permutation process as specified in section 7.13.2.2 with the input variable n.
2. If n is equal to 6, invoke B(32 + i, 63 - i, 63 - 4 * brev(4, i), 0, r) for i = 0..15.
3. If n is greater than or equal to 5, invoke B(16 + i, 31 - i, 6 + (brev(3, 7 - i) << 3), 0, r) for i = 0..7.
4. If n is equal to 6, invoke H(32 + i * 2, 33 + i * 2, i & 1, r) for i = 0..15.
5. If n is greater than or equal to 4, invoke B(8 + i, 15 - i, 12 + (brev(2, 3 - i) << 4), 0, r) for i = 0..3.
6. If n is greater than or equal to 5, invoke H(16 + 2 * i, 17 + 2 * i, i & 1, r) for i = 0..7.
7. If n is equal to 6, invoke B(62 - i * 4 - j, 33 + i * 4 + j, 60 - 16 * brev(2, i) + 64 * j, 1, r) for i = 0..3, for j = 0..1.
8. If n is greater than or equal to 3, invoke B(4 + i, 7 - i, 56 - 32 * i, 0, r) for i = 0..1.
9. If n is greater than or equal to 4, invoke H(8 + 2 * i, 9 + 2 * i, i & 1, r) for i = 0..3.
10. If n is greater than or equal to 5, invoke B(30 - 4 * i - j, 17 + 4 * i + j, 24 + (j << 6) + ((1 - i) << 5), 1, r) for i = 0..1, for j=0..1.
11. If n is equal to 6, invoke H(32 + i * 4 + j, 35 + i * 4 - j, i & 1, r) for i = 0..7, for j = 0..1.
12. Invoke B(2 * i, 2 * i + 1, 32 + 16 * i, 1 - i, r) for i = 0..1.
13. If n is greater than or equal to 3, invoke H(4 + 2 * i, 5 + 2 * i, i, r) for i = 0..1.
14. If n is greater than or equal to 4, invoke B(14 - i, 9 + i, 48 + 64 * i, 1, r) for i = 0..1.
15. If n is greater than or equal to 5, invoke H(16 + 4 * i + j, 19 + 4 * i - j, i & 1, r) for i = 0..3, for j = 0..1.
16. If n is equal to 6, invoke B(61 - i * 8 - j, 34 + i * 8 + j, 56 - i * 32 + (j >> 1) * 64, 1, r) for i = 0..1, for j = 0..3.
17. Invoke H(i, 3 - i, 0, r) for i = 0..1.
18. If n is greater than or equal to 3, invoke B(6, 5, 32, 1, r).
19. If n is greater than or equal to 4, invoke H(8 + 4 * i + j, 11 + 4 * i - j, i, r) for i = 0..1, for j = 0..1.
20. If n is greater than or equal to 5, invoke B(29 - i, 18 + i, 48 + (i >> 1) * 64, 1, r) for i = 0..3.
21. If n is equal to 6, invoke H(32 + 8 * i + j, 39 + 8 * i - j, i & 1, r) for i = 0..3, for j = 0..3.
22. If n is greater than or equal to 3, invoke H(i, 7 - i, 0, r) for i = 0..3.

	23. If n is greater than or equal to 4, invoke B(13 - i, 10 + i, 32, 1, r) for i = 0..1. 24. If n is greater than or equal to 5, invoke H(16 + i * 8 + j, 23 + i * 8 - j, i, r) for i = 0..1, for j = 0..3. 25. If n is equal to 6, invoke B(59 - i, 36 + i, i < 4 ? 48 : 112, 1, r) for i = 0..7. 26. If n is greater than or equal to 4, invoke H(i, 15 - i, 0, r) for i = 0..7. 27. If n is greater than or equal to 5, invoke B(27 - i, 20 + i, 32, 1, r) for i = 0..3. 28. If n is equal to 6, the following steps apply for i = 0..7: ◦ Invoke H(32 + i, 47 - i, 0, r). ◦ Invoke H(48 + i, 63 - i, 1, r). 29. If n is greater than or equal to 5, invoke H(i, 31 - i, 0, r) for i = 0..15. 30. If n is equal to 6, invoke B(55 - i, 40 + i, 32, 1, r) for i = 0..7. 31. If n is equal to 6, invoke H(i, 63 - i, 0, r) for i = 0..31. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 298-300 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10D] performing inverse-orthogonal transform to the post-	Users of the Samsung Accused Products perform: inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area;	For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec to perform an inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data (as described above with respect to [10C], which is incorporated here by reference) to produce a decoded residual picture of one video area (*i.e.*, residual or reconstructed prediction residue).  FIGURE 1. **Block diagram of a typical hybrid video encoder.** https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

The reconstruction and dequantization process is defined as follows:

The variable dqDenom is derived as follows:

- If txSz is equal to TX_32X32, TX_16X32, TX_32X16, TX_16X64, or TX_64X16, dqDenom is set equal to 2.

- Otherwise, if txSz is equal to TX_64X64, TX_32X64, or TX_64X32, dqDenom is set equal to 4.
- Otherwise, dqDenom is set equal to 1.

The variable log2W (specifying the base 2 logarithm of the width of the transform block) is set equal to Tx_Width_Log2[txSz].

The variable log2H (specifying the base 2 logarithm of the height of the transform block) is set equal to Tx_Height_Log2[txSz].

The variable w (specifying the width of the transform block) is set equal to 1 << log2W.

The variable h (specifying the height of the transform block) is set equal to 1 << log2H.

The variable tw is set equal to Min(32, w).

The variable th is set equal to Min(32, h).

The variable flipUD is derived as follows. If PlaneTxType is equal to one of FLIPADST_DCT, FLIPADST_ADST, V_FLIPADST, or FLIPADST_FLIPADST, flipUD is set equal to 1. Otherwise, flipUD is set equal to 0.

The variable flipLR is derived as follows. If PlaneTxType is equal to one of DCT_FLIPADST, ADST_FLIPADST, H_FLIPADST, or FLIPADST_FLIPADST, flipLR is set equal to 1. Otherwise, flipLR is set equal to 0.

The following ordered steps apply:

1. For i = 0..(th-1), for j = 0..(tw-1), the following ordered steps apply:

   a. The variable q is derived as follows:

      - If i is equal to 0 and j is equal to 0, the variable q is set equal to get_dc_quant(plane).
      - Otherwise (i, j or both are not equal to 0), the variable q is set equal to get_ac_quant(plane).

   b. The variable q2 is derived as follows:

      - If using_qmatrix is equal to 1, PlaneTxType is less than IDTX, and SegQMLevel[plane][segment_id] is less than 15, q2 is set equal to Round2(q * Quantizer_Matrix[SegQMLevel[plane][segment_id]][plane > 0][Qm_Offset[txSz] + i * tw + j], 5).
      - Otherwise, q2 is set equal to q.

   c. The variable dq is set equal to Quant[i * tw + j] * q2.

   d. The variable sign is set equal to (dq < 0) ? -1 : 1.

   e. The variable dq2 is set equal to sign * (Abs(dq) & 0xFFFFFF) / dqDenom.

   f. Dequant[i][j] is set equal to Clip3(- (1 << (7 + BitDepth)), (1 << (7 + BitDepth)) - 1, dq2).

2. Invoke the 2D inverse transform block process defined in section 7.13.3 with the variable txSz as input. The inverse transform outputs are stored in the Residual buffer.
3. For i = 0..(h-1), for j = 0..(w-1), the following applies:
    - The variable xx is set equal to flipLR ? (w - j - 1) : j.
    - The variable yy is set equal to flipUD ? (h - i - 1) : i.
    - CurrFrame[plane][y + yy][x + xx] is set equal to Clip1(CurrFrame[plane][y + yy][x + xx] + Residual[i][j]).

If Lossless is equal to 1, it is a requirement of bitstream conformance that the values written into the Residual array in step 2 are representable by a signed integer with 1 + BitDepth bits.

Note: When Lossless is equal to 0, values written into Residual may not be representable by 1 + BitDepth bits (for example, due to quantization noise). The constraints in other parts of the specification ensure that the values will always be representable by a signed integer with Max(BitDepth + 5, 15) bits.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288-296

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

	The input to this process is a variable txSz that specifies the transform size. Set the variable log2W equal to Tx_Width_Log2[txSz]. Set the variable log2H equal to Tx_Height_Log2[txSz]. Set the variable w equal to 1 << log2W. Set the variable h equal to 1 << log2H.

Set the variable rowShift equal to Lossless ? 0 : Transform_Row_Shift[txSz].

Set the variable colShift equal to Lossless ? 0 : 4.

Set the variable rowClampRange equal to BitDepth + 8.

Set the variable colClampRange equal to Max(BitDepth + 6, 16).

The row transforms with i = 0..(h-1) are applied as follows:

- T[j] is derived as follows for j = 0..(w-1):
    - If i and j are both less than 32, T[j] is set equal to Dequant[i][j].
    - Otherwise, T[j] is set equal to 0.
- If Abs(log2W - log2H) is equal to 1, T[j] is set equal to Round2(T[j] * 2896, 12) for j = 0..(w-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 2.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, ADST_DCT, FLIPADST_DCT or H_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, if PlaneTxType is equal to one of DCT_ADST, ADST_ADST, DCT_FLIPADST, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, H_ADST, or H_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2W and the input variable r equal to rowClampRange.
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2W.
- Set Residual[i][j] equal to Round2(T[j], rowShift) for j = 0..(w-1).

Between the row and column transforms, Residual[i][j] is set equal to Clip3(- (1 << (colClampRange - 1)), (1 << (colClampRange - 1)) - 1, Residual[i][j]) for i = 0..(h-1), for j = 0..(w-1).

The column transforms with j = 0..(w-1) are applied as follows:

- Set T[i] equal to Residual[i][j] for i = 0..(h-1).
- If Lossless is equal to 1, invoke the Inverse WHT process as specified in section 7.13.2.10 with shift equal to 0.
- Otherwise, if PlaneTxType is equal to one of DCT_DCT, DCT_ADST, DCT_FLIPADST or V_DCT, invoke the inverse DCT process as specified in section 7.13.2.3 with the input variable n equal to log2H and the input variable r equal to colClampRange.

- Otherwise, if PlaneTxType is equal to one of ADST_DCT, ADST_ADST, FLIPADST_DCT, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, V_ADST, or V_FLIPADST, invoke the inverse ADST process as specified in section 7.13.2.9 with input variable n equal to log2H and the input variable r equal to colClampRange.

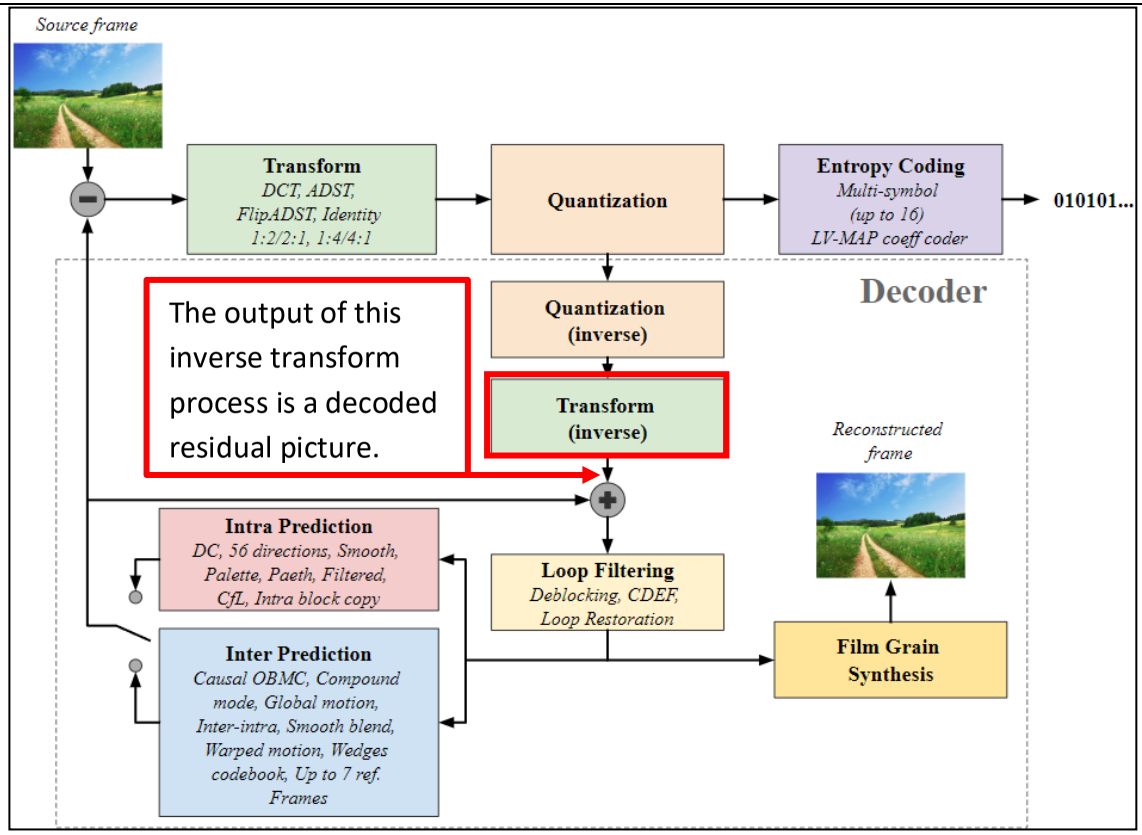
- Otherwise, invoke the inverse identity transform process specified in section 7.13.2.15 with the input variable n equal to log2H.

- Residual[i][j] is set equal to Round2(T[i], colShift) for i = 0..(h-1).

where Transform_Row_Shift is defined as:

```
Transform_Row_Shift[ TX_SIZES_ALL ] = {
  0, 1, 2, 2, 2, 0, 0, 1, 1,
  1, 1, 1, 1, 1, 1, 2, 2, 2, 2
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305-307

	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10E1] defining a boundary condition of a	Users of the Samsung Accused Products perform: defining a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data. Samsung

border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data, and	directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* elements [10A3] and [10A4], which are incorporated here by reference. Specifically, the Samsung 8K Neo QLED QN800A defines a boundary condition of a border that corresponds to the border motion vector data, from the reference picture based on the border motion-vector data through algorithms such as overlapped block motion compensation using the AV1 codec. A boundary condition of a reference picture border is defined corresponding to the border motion vector data because the boundary conditions (i.e., immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, which includes true motion trajectory data) correlates or corresponds to the motion vector data (i.e., motion vectors on both sides). Other prediction modes supported by AV1 that blend multiple reference blocks include an overlapped block motion compensation and a combined inter-intra prediction mode, both of which operate on a single reference frame and allow only one motion vector. **Overlapped Block Motion Compensation** The overlapped block motion compensation mode modifies the original design in [28] to account for variable block sizes [29]. It exploits the immediate spatial neighbors' motion information to improve the prediction quality for pixels near its top and left boundaries, where the true motion trajectory correlates with the motion vectors on both sides. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10E2] generate an estimated video signal in each rectangular zone in the picture to be decoded, that satisfies Poisson's Equation, thus producing a first predictive picture;	Users of the Samsung Accused Products perform: generating an estimated video signal in each rectangular zone in the picture to be decoded, that satisfies Poisson's Equation, thus producing a first predictive picture. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. *See* element [10A5], which is incorporated here by reference. Specifically, the Samsung 8K Neo QLED QN800A practices overlapped block motion compensation using the AV1 codec. The Overlapped Block Motion Compensation (OBMC) process in AV1 is used to generate an estimated video signal (*i.e.*, prediction block). The estimated video signal is generated in a multistep process that involves finding predicted pixels of a block in steady state (*i.e.*, that minimize the residual). In each step, pixels from a neighboring block are blended smoothly with the presently generated signal which produces an intermediate blending result. The blending is performed using a smooth weighting function to produce a predictive picture.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

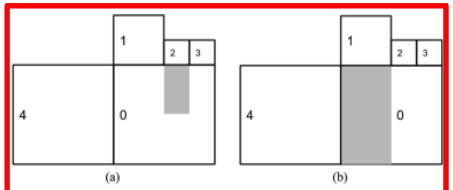


Fig. 1. Overlapping regions

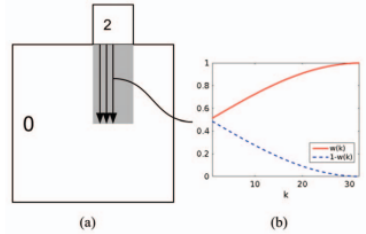


Fig. 2. Weighting based on 1-D raised cosine window

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

7.11.3.9. Overlapped motion compensation process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables w and h specifying the width and height of the region to be predicted.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

This process blends the inter predicted samples for the current block with inter predicted samples based on motion vectors from the above and left blocks.

The maximum number of overlapped predictions is limited based on the size of the block.

For small blocks, only the left neighbor will be used to form the prediction.

The variables subX and subY describing the subsampling of the current plane are derived as follows:

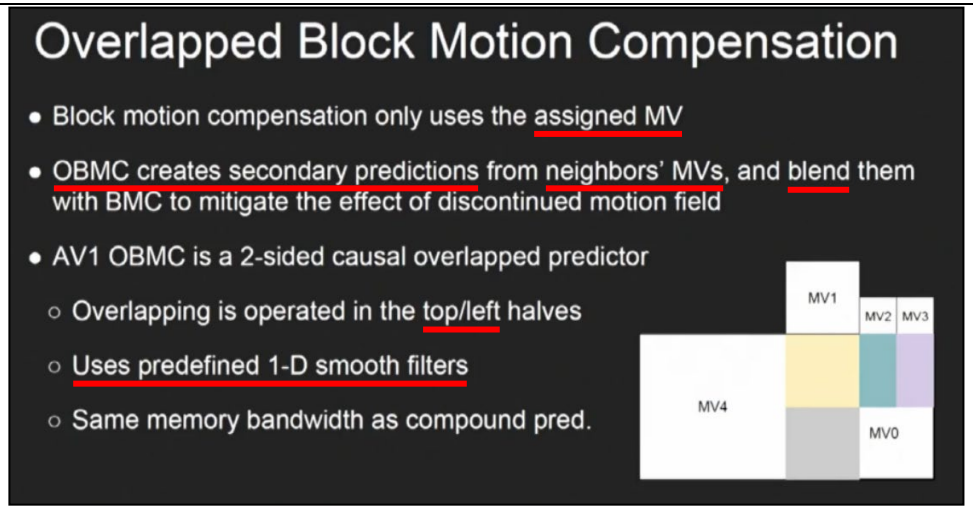
- If plane is equal to 0, subX and subY are set equal to 0.
- Otherwise (plane is not equal to 0), subX is set equal to subsampling_x and subY is set equal to subsampling_y.

The process is specified as:

```
if ( AvailU ) {
    if ( get_plane_residual_size( MiSize, plane ) >= BLOCK_8X8 ) {
        pass = 0
        w4 = Num_4x4_Blocks_Wide[ MiSize ]
        x4 = MiCol
        y4 = MiRow
        nCount = 0
        nLimit = Min(4, Mi_Width_Log2[ MiSize ])
        while ( nCount < nLimit && x4 < Min( MiCols, MiCol + w4 ) ) {
            candRow = MiRow - 1
            candCol = x4 | 1
            candSz = MiSizes[ candRow ][ candCol ]
            step4 = Clip3( 2, 16, Num_4x4_Blocks_Wide[ candSz ] )
            if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
                nCount += 1
                predW = Min( w, ( step4 * MI_SIZE ) >> subX )
                predH = Min( h >> 1, 32 >> subY )
                mask = get_obmc_mask( predH )
                predict_overlap( )
            }
            x4 += step4
        }
    }
}
if ( AvailL ) {
    pass = 1
    h4 = Num_4x4_Blocks_High[ MiSize ]
    x4 = MiCol
    y4 = MiRow
    nCount = 0
    nLimit = Min(4, Mi_Height_Log2[ MiSize ])
    while ( nCount < nLimit && y4 < Min( MiRows, MiRow + h4 ) ) {
        candCol = MiCol - 1
        candRow = y4 | 1
        candSz = MiSizes[ candRow ][ candCol ]
        step4 = Clip3( 2, 16, Num_4x4_Blocks_High[ candSz ] )
        if ( RefFrames[ candRow ][ candCol ][ 0 ] > INTRA_FRAME ) {
            nCount += 1
            predW = Min( w >> 1, 32 >> subX )
            predH = Min( h, ( step4 * MI_SIZE ) >> subY )
            mask = get_obmc_mask( predW )
            predict_overlap( )
        }
        y4 += step4
    }
}
```

When the function predict_overlap is invoked, the following ordered steps apply to form the overlap prediction for a region of size predW by predH based on the candidate motion vector:

	1. The motion vector mv is set equal to Mvs[candRow][candCol][0]. 2. The variable refIdx is set equal to ref_frame_idx[RefFrames[candRow][candCol][0] - LAST_FRAME]. 3. The variable predX is set equal to (x4 * 4) >> subX. 4. The variable predY is set equal to (y4 * 4) >> subY. 5. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, predX, predY, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY. 6. The block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, predW, predH, candRow, candCol as inputs and the output is assigned to the 2D array obmcPred. 7. obmcPred[i][j] is set equal to Clip1(obmcPred[i][j]) for i = 0..predH-1 and j = 0..predW-1. 8. The blending process in section 7.11.3.10 is invoked with plane, predX, predY, predW, predH, pass, obmcPred, and mask as inputs. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 275-277 Block motion compensation uses assigned motion vectors (MVs) and then OBMC creates secondary predictions from neighbor MVs in the overlapped top and left directions and then blends them for smoothness. The process involves finding predicted pixels of a block in steady state (that minimize the residual).



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.11.3.10. Overlap blending process

The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables predX and predY specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables predW and predH specifying the width and height of the region to be predicted,
- a variable pass equal to 0 if blending above samples, or equal to 1 if blending left samples,
- a 2d array obmcPred containing the samples predicted from a neighboring motion vector,
- an array mask containing the blending weights.

The outputs of this process are modified inter predicted samples in the current frame CurrFrame.

For i = 0..(predH - 1) and j = 0..(predW - 1), the following ordered steps apply:

1. The variable m specifying the blending factor is specifed as follows:
    - If pass is equal to 0 (blend from above), m is set equal to mask[i].
    - Otherwise (pass is equal to 1 meaning blend from left), m is set equal to mask[j].
2. CurrFrame[plane][predY + i][predX + j] is set equal to Round2(m * CurrFrame[plane][predY + i][predX + j] + (64 - m) * obmcPred[i][j], 6)

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 278

Additionally, the AV1 decoder generates an estimated video signal in each rectangular zone in the picture to be coded (*i.e.,* each prediction block $p_{OBMC}(x,y)$) that satisfies Poisson's Equation. The general form of Poisson's Equation is shown as: $\Delta u(x,y)=f(x,y)$. Δ is the Laplacian function, $u(x,y)$ is some function that models the changes in a given phenomenon, and $f(x,y)$ is a source function that influences the phenomenon over a given domain Ω. For generating the estimated signal, the domain of interest, Ω, is the current block being predicted. As indicated in the highlighted equations below, the algorithm for calculating the estimated video signal (*i.e.*, each predicted block $p_{OBMC}(x,y)$) utilizes the form of the Poisson Equation.

2.1. Causal OBMC for Variable Block-size Coding

This algorithm starts with initializing the overlapped prediction $p_{obmc}(x, y)$ as the basic prediction by applying the motion assigned to the current block. For an $N \times M$ prediction unit, let $p_i(x, y)$ $(0 \leq x < M, 0 \leq y < N)$ denote a prediction block for pixels in the current block $(i = 0)$ but computed using the reference and motion vector combination(s) of block i, so $p_{obmc}(x, y)$ is initialized as $p_0(x, y)$. Note that both compound and single reference frame predictors are allowed to contribute to OBMC, so p_i could be a combined prediction, which averages two single-reference predictions in the AV1 codec.

Next, prediction overlapping will be performed progressively on top of the continuously updated $p_{obmc}(x, y)$ in two stages: first with the adjacent blocks above, and next with those to the left. For example, in Fig.1, block 0 is the unit to predict, block 1-4 are coded adjacent blocks. At the first stage, we check the predictors of block 1-3, located right on top of block 0. Among them, if block i is coded by an inter predictor, the assigned motion parameters are exploited to compute a new prediction block $p_i(x, y)$ for the 'overlapping region' of p_0 and p_i in block 0: a rectangular area in block 0, with the common edge of block 0 and block i as one of its edges and the opposing edge on the mid-line of block 0. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2 for pixels in block 0, and the predictor blending will only be operated there. A K-tap 1-D filter $w_K(k)$, which is shaped as a half raised cosine window

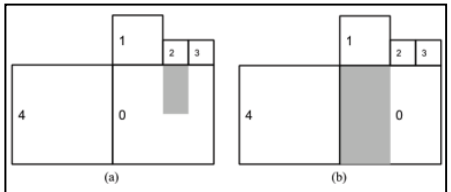


Fig. 1. Overlapping regions

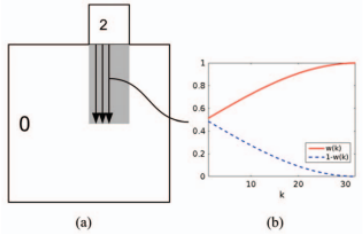


Fig. 2. Weighting based on 1-D raised cosine window

424

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2} sin(\frac{\pi}{2K}(k + \frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K - 1 \quad (1)$$

is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

https://ieeexplore.ieee.org/document/8296419

their scan order. The motion vector of each selected reference block is employed to generate a motion-compensated block that extends from the top boundary toward the center of the current block. Its width is the same as the reference block's width, and its height is half of the current block's height, as shown in Fig. 14(a). An intermediate blending result is formed as

$$P_{\text{int}}(x,y) = m(x,y)R_1(x,y) + (64 - m(x,y))R_{\text{above}}(x,y) \tag{15}$$

where $R_1(x,y)$ is the original motion-compensated pixel at position (x,y) using current block's motion vector mv_0, and $R_{\text{above}}(x,y)$ is the pixel from the overlapped reference block. The weight $m(x,y)$ follows a raised cosine function:

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{H}\left(y+\frac{1}{2}\right)\right)+\frac{1}{2}\right) \tag{16}$$

where $y = 0, 1, \ldots, H/2 - 1$ is the row index, and H is the current block height. The weight distribution for $H = 16$ is shown in Fig. 15.

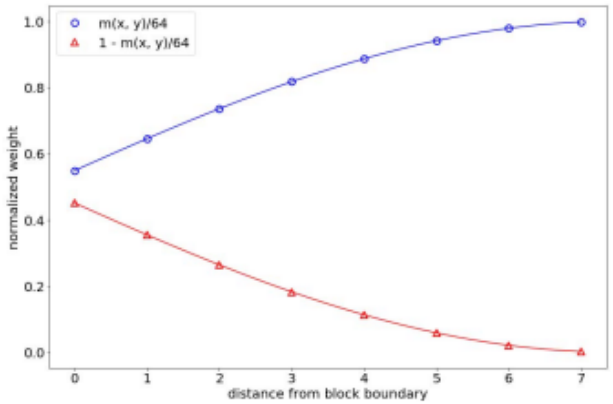


Fig. 15. ***Normalized weights for OBMC with H = 16 or W = 16.***

The scheme next processes the immediate left neighbors to extract the available motion vectors and build overlapped reference blocks extending from the left boundary toward the center, as shown in Fig. 14(b). The final prediction is calculated by

$$P(x,y) = m(x,y)P_{\text{int}}(x,y) + (64 - m(x,y))R_{\text{left}}(x,y) \quad (17)$$

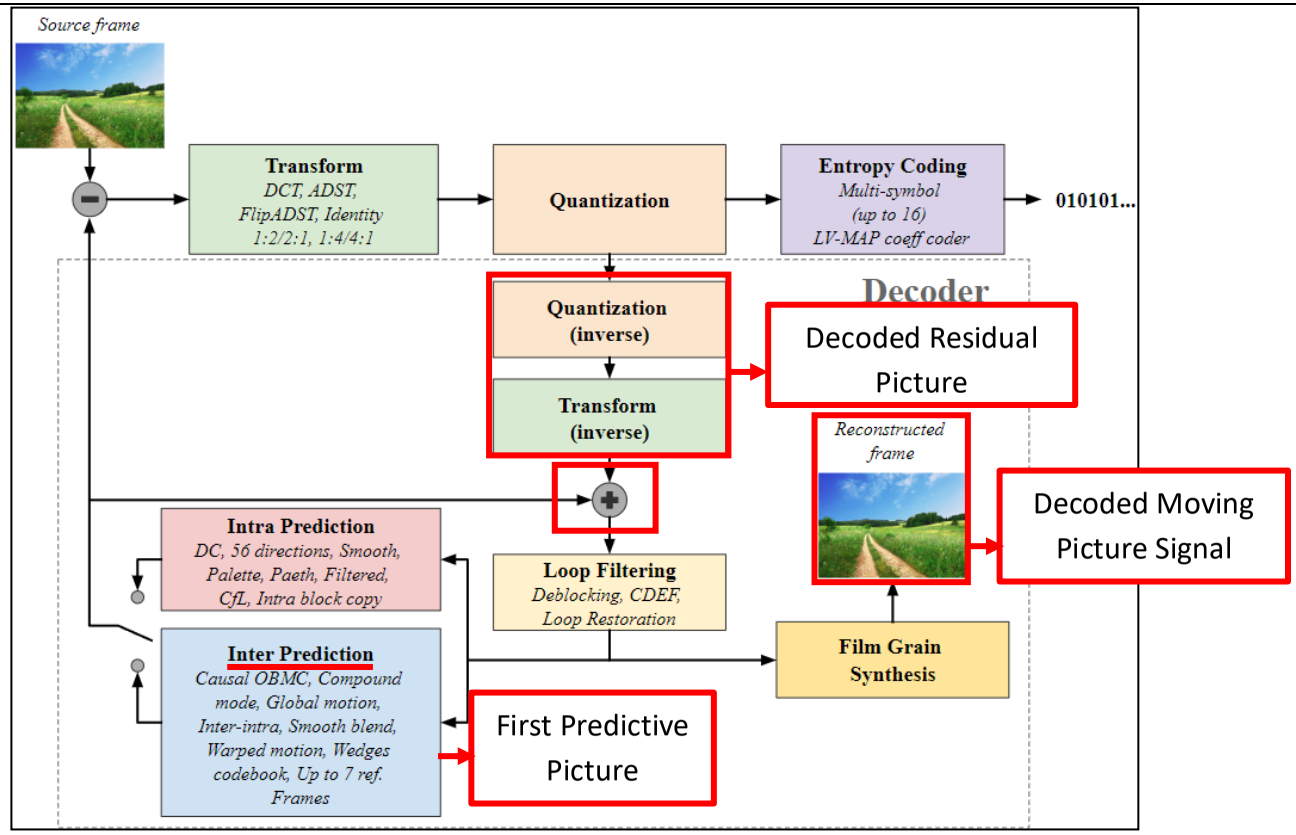
where $R_{\text{left}}(x,y)$ is the pixel from the left-hand side overlapped reference block. The weight $m(x,y)$ is a raised cosine function of the column index x

$$m(x,y) = 64 * \left(\frac{1}{2}\sin\left(\frac{\pi}{W}\left(x + \frac{1}{2}\right)\right) + \frac{1}{2}\right) \quad (18)$$

where $x = 0, 1, \ldots, W/2 - 1$ and W is the current block width.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[10F] combining the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal; and	Users of the Samsung Accused Products perform: combining the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A combines the predictive picture and the decoded residual picture to generate a decoded moving-picture signal using the AV1 codec. Further, to generate a decoded moving picture signal (*i.e.*, the reconstructed frame or reconstructed predicted samples), the AV1 codec combines (*i.e.*, reconstructs) the first predictive picture (*e.g.*, the inter predicted picture or corresponding prediction values) and the decoded residual picture (*i.e.,* the decoded residual, the inverse quantized and inverse transformed picture, or reconstructed prediction residue). **Reconstruction** Obtaining the addition of the decoded residual and the corresponding prediction values. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 4



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

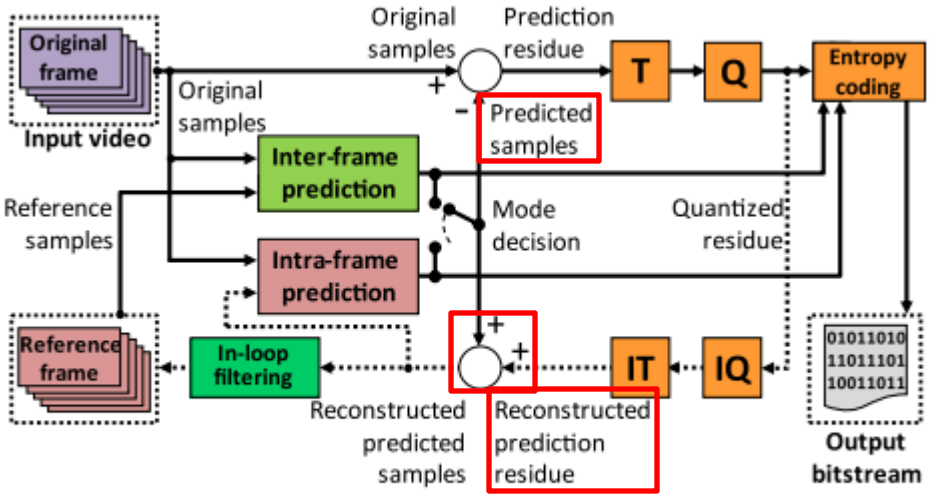


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

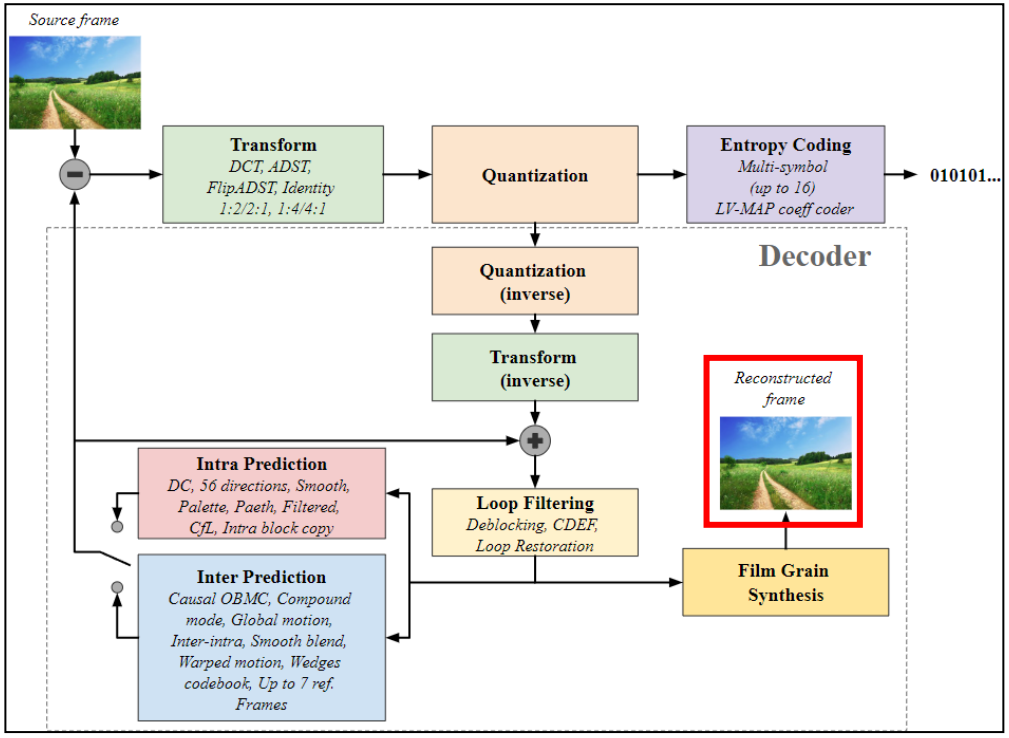
https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 288

7.13.3. 2D inverse transform process

This process performs a 2D inverse transform for an array of coefficients stored in the 2D array Dequant. The output is placed in the 2D array Residual.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 305

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[10G] storing the decoded moving-picture signal for at least one picture as a reference picture.	Users of the Samsung Accused Products perform: storing the decoded moving-picture signal for at least one picture as a reference picture. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses its AV1 codec to store the decoded moving-picture signal (*i.e.*, decoded samples for the current frame) as reference frames (*i.e.*, an updated set of reference frames) and each of its parameters in memory.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.20. Reference frame update process

This process is invoked as the final step in decoding a frame.

The inputs to this process are the decoded samples for the current frame LrFrame[plane][x][y].

The output from this process is an updated set of reference frames and previous motion vectors.

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefValid[i] is set equal to 1.
- RefFrameId[i] is set equal to current_frame_id.
- RefUpscaledWidth[i] is set equal to UpscaledWidth.
- RefFrameWidth[i] is set equal to FrameWidth.
- RefFrameHeight[i] is set equal to FrameHeight.
- RefRenderWidth[i] is set equal to RenderWidth.
- RefRenderHeight[i] is set equal to RenderHeight.

	• RefMiCols[i] is set equal to MiCols. • RefMiRows[i] is set equal to MiRows. • RefFrameType[i] is set equal to frame_type. • RefSubsamplingX[i] is set equal to subsampling_x. • RefSubsamplingY[i] is set equal to subsampling_y. • RefBitDepth[i] is set equal to BitDepth. • SavedOrderHints[i][j + LAST_FRAME] is set equal to OrderHints[j + LAST_FRAME] for j = 0..REFS_PER_FRAME-1. • FrameStore[i][0][y][x] is set equal to LrFrame[0][y][x] for x = 0..UpscaledWidth-1, for y = 0..FrameHeight-1. • FrameStore[i][plane][y][x] is set equal to LrFrame[plane][y][x] for plane = 1..2, for x = 0..((UpscaledWidth + subsampling_x) >> subsampling_x) - 1, for y = 0..((FrameHeight + subsampling_y) >> subsampling_y) - 1. • SavedRefFrames[i][row][col] is set equal to MfRefFrames[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1. • SavedMvs[i][row][col][comp] is set equal to MfMvs[row][col][comp] for comp = 0..1, for row = 0..MiRows-1, for col = 0..MiCols-1. • SavedGmParams[i][ref][j] is set equal to gm_params[ref][j] for ref = LAST_FRAME..ALTREF_FRAME, for j = 0..5. • SavedSegmentIds[i][row][col] is set equal to SegmentIds[row][col] for row = 0..MiRows-1, for col = 0..MiCols-1. • The function save_cdfs(i) is invoked (see below). • If film_grain_params_present is equal to 1, the function save_grain_params(i) is invoked (see below). • The function save_loop_filter_params(i) is invoked (see below). • The function save_segmentation_params(i) is invoked (see below).

For each value of i from 0 to NUM_REF_FRAMES - 1, the following applies if bit i of refresh_frame_flags is equal to 1 (i.e. if (refresh_frame_flags >> i) & 1 is equal to 1):

- RefOrderHint[i] is set equal to OrderHint.

save_cdfs(ctx) is a function call that indicates that all the CDF arrays are saved into frame context number ctx in the range 0 to (NUM_REF_FRAMES - 1). When this function is invoked the following takes place:

- A copy of each CDF array mentioned in the semantics for init_coeff_cdfs and init_non_coeff_cdfs is saved in an area of memory indexed by ctx.

save_grain_params(i) is a function call that indicates that all the syntax elements that can be read in film_grain_params should be saved into an area of memory indexed by i.

save_loop_filter_params(i) is a function call that indicates that the values of loop_filter_ref_deltas[j] for j = 0 .. TOTAL_REFS_PER_FRAME-1, and the values of loop_filter_mode_deltas[j] for j = 0 .. 1 should be saved into an area of memory indexed by i.

save_segmentation_params(i) is a function call that indicates that the values of FeatureEnabled[j][k] and FeatureData[j][k] for j = 0 .. MAX_SEGMENTS-1, for k = 0 .. SEG_LVL_MAX-1 should be saved into an area of memory indexed by i.

Note: Although this process stores all the motion vectors into SavedMvs, only the values where row and col are both odd will affect the decoding process.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 349-351

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claims 1-4 (the "Asserted Claims") of U.S. Patent No. 8,139,150 (the "'150 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung 8K Neo QLED QN800A television, depicted in this chart, is believed to be representative of other televisions and display products, including at least the DBJ Series, QN900A, QN800A, QN700A, Q950T, Q900T, Q800T, Q700T, QN95A, QN91A, QN90A, QN85A, Q80A, Q70A, Q95T, Q90T, Q80T, Q70T, Q60T, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, and TU7000 for the purposes of infringement of the '150 Patent. Based on publicly available information, ACT believes that the Samsung 8K Neo QLED QN800A television, charted herein, and other televisions/display products share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the listed televisions/display products all purport to support H.264 Multi View Coding (MVC). Should discovery yield documents or information demonstrating that Defendants sell other infringing products which are not represented by the Samsung 8K Neo QLED QN800A television, ACT reserves its right to amend its infringement contentions to include a chart mapping the claims of the '150 Patent to the structure and function of the newly-discovered products.

Samsung directly infringes each of the Asserted Claims by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

Samsung indirectly infringes the Asserted Claims in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through their operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claims; and (ii) marketing and advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers,

including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claims, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claims but has taken deliberate actions to avoid learning these facts.

Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

ACT does not concede that any claims of the '150 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

<table>
<tr><th>Claim – 8,139,150</th><th>Exemplary Supporting Evidence Regarding Samsung's Accused Products</th></tr>
<tr><td>[1Pa] An apparatus for decoding a coded multi-view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints respectively,</td><td>The Samsung Accused Products comprise: An apparatus for decoding a coded multi-view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints, respectively. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC).

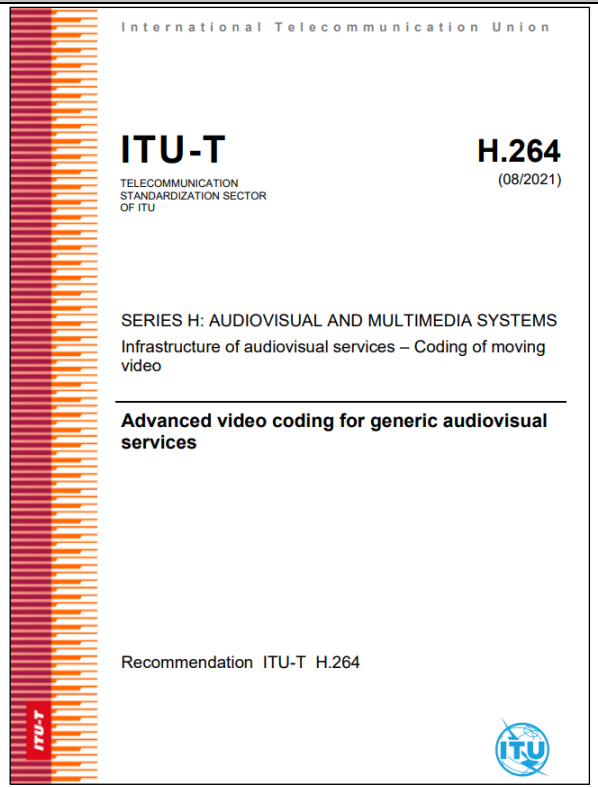
This element is infringed literally, or in the alternative, under the doctrine of equivalents.

For example, the Samsung 8K Neo QLED QN800A television supports the H.264 MVC codec.

Supported video codecs (QN7**A Series or higher)
<table>
<tr><th>File format</th><th>Container</th><th>Video codecs</th><th>Resolution</th><th>Frame rate (fps)</th><th>Bitrate (Mbps)</th><th>Audio codecs</th></tr>
<tr><td>*.avi
*.mkv
*.asf
*.wmv
*.mp4
*.mov
*.3gp
*.vro
*.mpg
*.mpeg
*.ts
*.tp
*.trp
*.flv
*.vob
*.svi
*.m2ts
*.mts</td><td>AVI
MKV
ASF
MP4
3GP
MOV
FLV
VRO
VOB
PS
TS
SVAF</td><td>H.264 BP/MP/HP</td><td>3840 x 2160</td><td>3840 x 2160: 60
1920 x 1080: 120</td><td>60</td><td rowspan="15">Dolby Digital
LPCM
ADPCM(IMA, MS)
AAC
HE-AAC
WMA
Dolby Digital+
MPEG(MP3)
AC-4
G.711(A-Law, μ-Law)
OPUS</td></tr>
<tr><td>HEVC (H.265 - Main, Main10)</td><td>7680 x 4320</td><td>7680 x 4320: 60
3840 x 2160: 120</td><td>100</td></tr>
<tr><td>Motion JPEG</td><td>3840 x 2160</td><td>30</td><td>80</td></tr>
<tr><td>MVC</td><td rowspan="11">1920 x 1080</td><td rowspan="4">60</td><td rowspan="11">20</td></tr>
<tr><td>MPEG4 SP/ASP</td></tr>
<tr><td>Window Media Video v9 (VC1)</td></tr>
<tr><td>MPEG2</td></tr>
<tr><td>MPEG1</td><td rowspan="7">30</td></tr>
<tr><td>Microsoft MPEG-4 v1, v2, v3</td></tr>
<tr><td>Window Media Video v7 (WMV1), v8 (WMV2)</td></tr>
<tr><td>H.263 Sorrenson</td></tr>
<tr><td>VP6</td></tr>
<tr><td>AV1</td><td>7680 x 4320</td><td>7680 x 4320: 60
3840 x 2160: 120</td><td>80</td></tr>
</table></td></tr>
</table>

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf The Samsung 8K Neo QLED QN800A is an apparatus for decoding a coded multi-view video signal (*e.g.*, a coded video sequence conforming to an MVC profile specified in Annex H) including coded signals resulting from encoding picture signals corresponding to different viewpoints (*e.g.*, view components) respectively. H.264 Annex H provides the definition of coding for a multi-view video signal, supporting coding of different viewpoints, including binocular 3D or binocular stereoscopy, as described in the '150 Patent. H.264 Annex C describes the structure and operation of a reference decoder to support the coding standard.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products



International Telecommunication Union

ITU-T **H.264**

(08/2021)

TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving video

Advanced video coding for generic audiovisual services

Recommendation ITU-T H.264

The 4th edition of ITU-T H.264, approved in 2009-05, contained enhancement extensions to support multiview video coding (MVC), specification of a "Constrained Baseline Profile", and some miscellaneous corrections and clarifications.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Annex H

Multiview video coding

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies multiview video coding, referred to as MVC.

H.1 Scope

Bitstreams and decoders conforming to the profile specified in this annex are completely specified in this annex with reference made to clauses 2 to 9 and Annexes A to E.

H.3.46 **view:** A sequence of *view components* associated with an identical value of view_id.

H.3.47 **view component:** A *coded representation* of a *view* in a single *access unit*. When profile_idc is equal to 134, a view contains samples of two distinct spatially packed constituent frames that are packed into one frame using one of the frame packing arrangement schemes as specified in clause D.2.26.

Annex C

Hypothetical reference decoder

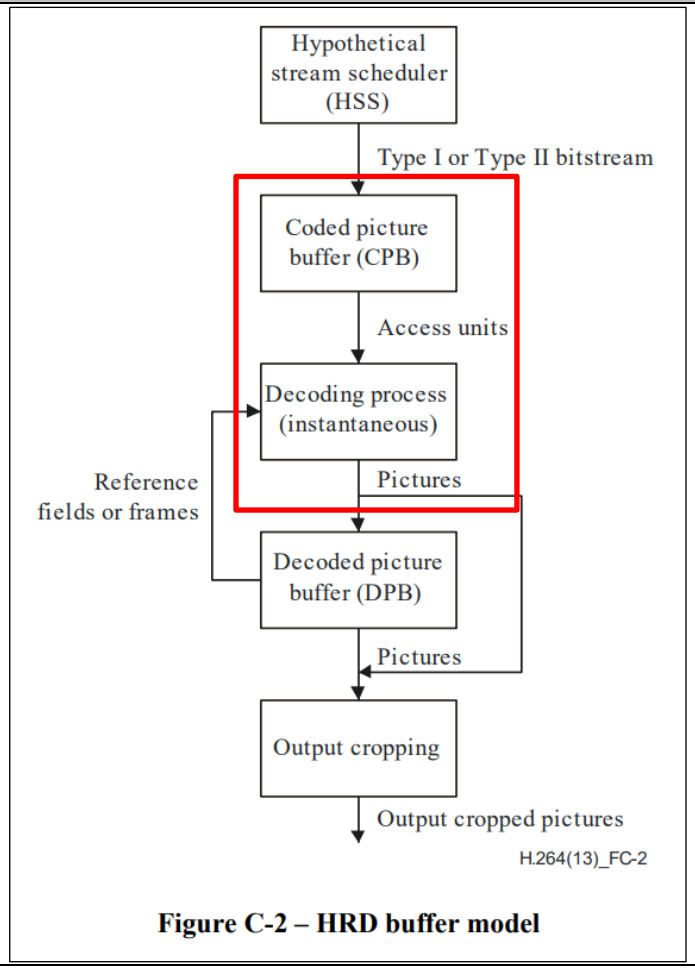
(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1Pb] wherein at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint, and	In the Samsung Accused Products, at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, in multi-view decoding, at least one among the viewpoints is designated as a base viewpoint (*e.g.*, base view), the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints (*e.g.*, independently of other views) except the base viewpoint. For example, the base view can be decoded independently (without referring to) any other view and there is only one base view in a coded video sequence. **H.3.6** **base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1Pc] wherein the coded multi-view video signal contains coded information representing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint, the apparatus comprising:	In the Samsung Accused Products, the coded multi-view video signal contains coded information representing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, the coded multi-view video signal (*e.g.*, MVC coded video sequence) contains coded information representing a desired delay time (*e.g.*, *cpb_removal_delay*, picture timing SEI messages that are included in MVC scalable nesting Supplemental Enhanced Information (SEI)messages) of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. The time at which a coded picture for any view (an access unit) is decoded is determined by the coded picture buffer (CPB) removal time. For MVC coded pictures, this removal time is coded in nested SEI messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 **Figure C-2 – HRD buffer model**

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (a) When an access unit contains one or more buffering period SEI messages that are included in MVC scalable nesting SEI messages, the buffering period SEI message that is associated with the operation point being decoded is the buffering period SEI message that initializes the HRD. Let hrdVId[i] be equal to sei_op_view_id[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and let hrdTId be the value of sei_op_temporal_id, that are associated with the MVC scalable nesting SEI message containing the buffering period SEI message that initializes the HRD. (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. (c) The HRD parameters that are used for conformance checking are the HRD parameters included in the MVC video usability information extension of the active MVC sequence parameter set that are associated with values of vui_mvc_view_id[i][j] for all j in the range of 0 to vui_mvc_num_target_output_views_minus1[i], inclusive, equal to hrdVId[j], and the value of vui_mvc_temporal_id[i] equal to hrdTId. For the specification in this annex, num_units_in_tick, time_scale, fixed_frame_rate_flag, nal_hrd_parameters_present_flag, vcl_hrd_parameters_present_flag, low_delay_hrd_flag, and pic_struct_present_flag are substituted with the values of vui_mvc_num_units_in_tick[i], vui_mvc_time_scale[i], vui_mvc_fixed_frame_rate_flag[i], vui_mvc_nal_hrd_parameters_present_flag[i], vui_mvc_vcl_hrd_parameters_present_flag[i], vui_mvc_low_delay_hrd_flag[i], and vui_mvc_pic_struct_present_flag[i], respectively, with i being the value for which vui_mvc_view_id[i] is equal to hrdVId[j] for all j in the range of 0 to vui_mvc_num_traget_output_views_minus1[i], inclusive, and vui_mvc_temporal_id[i] equal to hrdTId.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	cpb_removal_delay specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1A] first means for decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints	The Samsung Accused Products comprise: first means for decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises first means (*e.g.*, hypothetical reference decoder (HRD)) for decoding the coded information in the coded multi-view video signal to recover the desired delay times (*e.g.*, *cpb_removal_delay*) for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. The *cpb_removal_delay* value must be decoded in order to determine when to remove the access unit from the CPB.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
except the base viewpoint;	**Annex C** **Hypothetical reference decoder** (This annex forms an integral part of this Recommendation | International Standard.) This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1B] second means for decoding the coded signal originating from the picture	The Samsung Accused Products comprise: second means for decoding the coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
signal corresponding to the base viewpoint;	For example, the Samsung 8K Neo QLED QN800A comprises second means (*e.g.*, MVC decoder) for decoding the coded signal originating from the picture signal corresponding to the base viewpoint. As the only view that can be independently decoded, and having the minimum index, the base view will be decoded first. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1C] third means for decoding the coded signals originating from the picture signals	The Samsung Accused Products comprise: third means for decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
corresponding to the viewpoints except the base viewpoint; and	For example, the Samsung 8K Neo QLED QN800A comprises third means (*e.g.*, MVC decoder) for decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. For example, other views in the coded sequence will be released from the CPB and instantaneously decoded subsequent to the base view, based on the view index. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1D] fourth means for delaying the decoding of the coded signals by the third means relative to the decoding of the coded signal by the second	The Samsung Accused Products comprise: fourth means for delaying the decoding of the coded signals by the third means relative to the decoding of the coded signal by the second means by time intervals equal to the desired delay times recovered by the first means. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. With respect to "the coded signals by the third means," the support for Claim [1C] is incorporated by reference. With respect to "the coded signal by the second means," the support for Claim [1B] is incorporated by reference.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
means by time intervals equal to the desired delay times recovered by the first means.	With respect to "the desired delay times recovered by the first means," the support for Claim [1A] is incorporated by reference. For example, the Samsung 8K Neo QLED QN800A comprises fourth means (*e.g.*, hypothetical reference decoder (HRD)) for delaying the decoding of the coded signals (*e.g.*, other viewpoints) by the third means (*e.g.*, MVC decoder) relative to (*e.g.* after) the decoding of the coded signal (*e.g.*, base view) by the second means (*e.g.*, MVC decoder) by time intervals equal to the desired delay times (*e.g.*, *cpb_removal_delay*) recovered by the first means (*e.g.*, hypothetical reference decoder). The time at which a coded picture for any view (an access unit) is decoded is determined by the CPB removal time. For MVC coded pictures, this removal time is coded in nested Supplemental Enhanced Information (SEI) messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

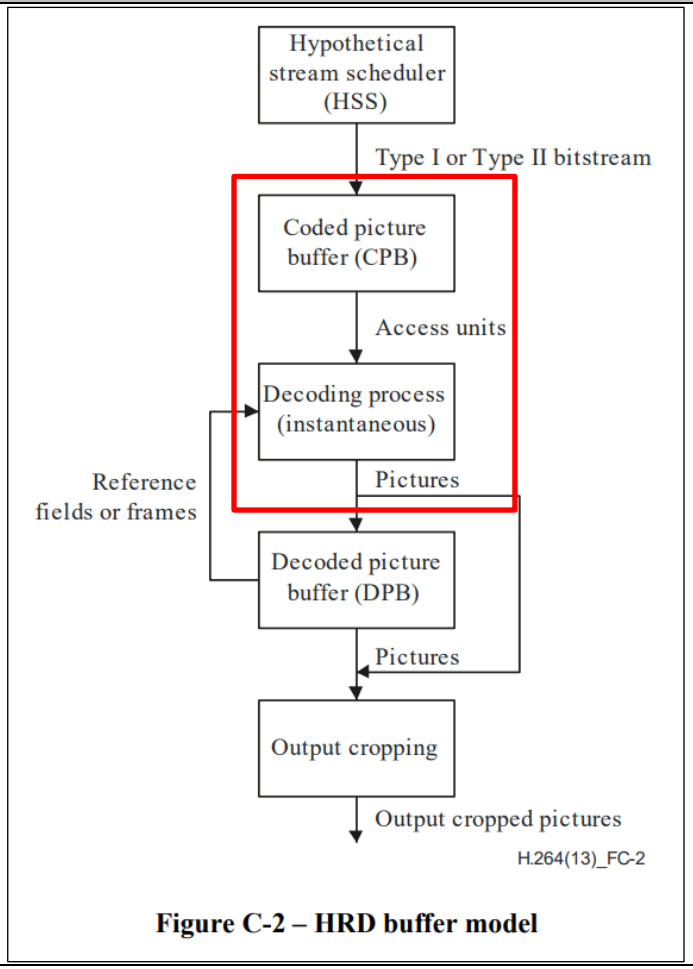


Figure C-2 – HRD buffer model

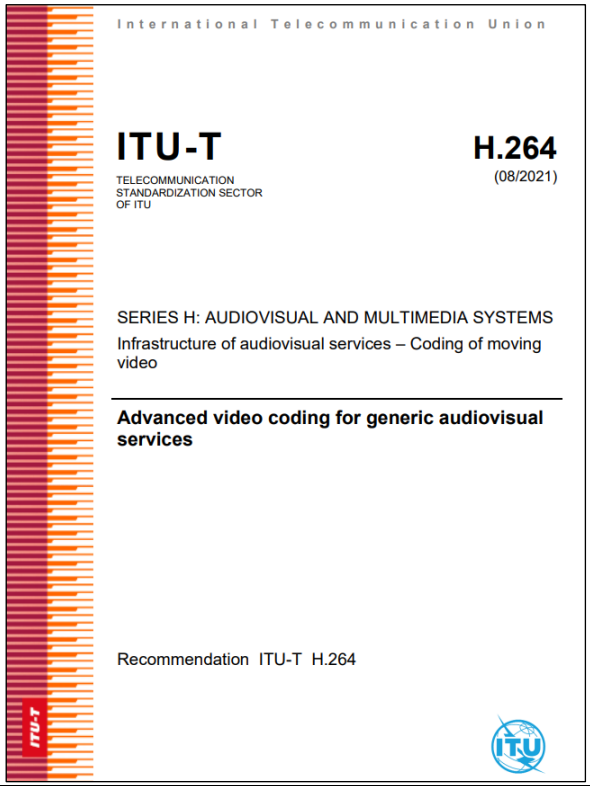
Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2Pa] A method of decoding a coded multi-	Users of the Samsung Accused Products perform a method of decoding a coded multi-view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints, respectively.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints respectively,	Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A television supports the H.264 MVC codec.

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
*.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The Samsung 8K Neo QLED QN800A is an apparatus for decoding a coded multi-view video signal (*e.g.*, a coded video sequence conforming to an MVC profile specified in Annex H) including coded signals resulting from encoding picture signals corresponding to different viewpoints (*e.g.*, view components) respectively. H.264 Annex H provides the definition of coding for a multi-view video signal, supporting coding of different viewpoints, including binocular 3D or binocular stereoscopy, as described in the '150 Patent. H.264 Annex C describes the structure and operation of a reference decoder to support the coding standard.  International Telecommunication Union ITU-T TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU H.264 (08/2021) SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving video Advanced video coding for generic audiovisual services Recommendation ITU-T H.264

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The 4th edition of ITU-T H.264, approved in 2009-05, contained enhancement extensions to support multiview video coding (MVC), specification of a "Constrained Baseline Profile", and some miscellaneous corrections and clarifications. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Annex H

Multiview video coding

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies multiview video coding, referred to as MVC.

H.1 Scope

Bitstreams and decoders conforming to the profile specified in this annex are completely specified in this annex with reference made to clauses 2 to 9 and Annexes A to E.

H.3.46 **view:** A sequence of *view components* associated with an identical value of view_id.

H.3.47 **view component:** A *coded representation* of a *view* in a single *access unit*. When profile_idc is equal to 134, a view contains samples of two distinct spatially packed constituent frames that are packed into one frame using one of the frame packing arrangement schemes as specified in clause D.2.26.

Annex C

Hypothetical reference decoder

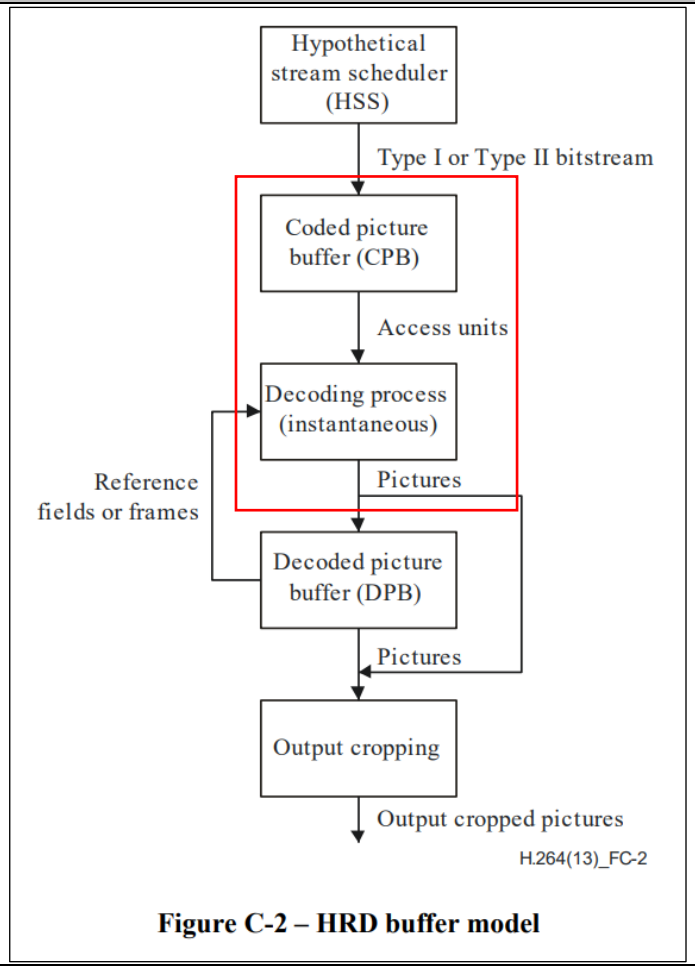
(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2Pb] wherein at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint, and	Users of the Samsung Accused Products perform a method wherein at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, in multi-view decoding, at least one among the viewpoints is designated as a base viewpoint (*e.g.*, base view), the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints (*e.g.*, independently of other views) except the base viewpoint. For example, the base view can be decoded independently (without referring to) any other view and there is only one base view in a coded video sequence. **H.3.6** **base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[2Pc] wherein the coded multi-view video signal contains coded information representing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint, the method	Users of the Samsung Accused Products perform a method wherein the coded multi-view video signal contains coded information representing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, the coded multi-view video signal (*e.g.*, MVC coded video sequence) contains coded information representing a desired delay time (*e.g.*, *cpb_removal_delay*, picture timing SEI messages that are included in MVC scalable nesting Supplemental Enhanced Information (SEI)messages) of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. The time at which a coded picture for any view (an access unit) is decoded is determined by the coded picture buffer (CPB) removal time. For MVC coded pictures, this removal time is coded in nested SEI messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
comprising the steps of:	 **Figure C-2 – HRD buffer model**

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (a) When an access unit contains one or more buffering period SEI messages that are included in MVC scalable nesting SEI messages, the buffering period SEI message that is associated with the operation point being decoded is the buffering period SEI message that initializes the HRD. Let hrdVId[i] be equal to sei_op_view_id[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and let hrdTId be the value of sei_op_temporal_id, that are associated with the MVC scalable nesting SEI message containing the buffering period SEI message that initializes the HRD. (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. (c) The HRD parameters that are used for conformance checking are the HRD parameters included in the MVC video usability information extension of the active MVC sequence parameter set that are associated with values of vui_mvc_view_id[i][j] for all j in the range of 0 to vui_mvc_num_target_output_views_minus1[i], inclusive, equal to hrdVId[j], and the value of vui_mvc_temporal_id[i] equal to hrdTId. For the specification in this annex, num_units_in_tick, time_scale, fixed_frame_rate_flag, nal_hrd_parameters_present_flag, vcl_hrd_parameters_present_flag, low_delay_hrd_flag, and pic_struct_present_flag are substituted with the values of vui_mvc_num_units_in_tick[i], vui_mvc_time_scale[i], vui_mvc_fixed_frame_rate_flag[i], vui_mvc_nal_hrd_parameters_present_flag[i], vui_mvc_vcl_hrd_parameters_present_flag[i], vui_mvc_low_delay_hrd_flag[i], and vui_mvc_pic_struct_present_flag[i], respectively, with i being the value for which vui_mvc_view_id[i] is equal to hrdVId[j] for all j in the range of 0 to vui_mvc_num_traget_output_views_minus1[i], inclusive, and vui_mvc_temporal_id[i] equal to hrdTId.

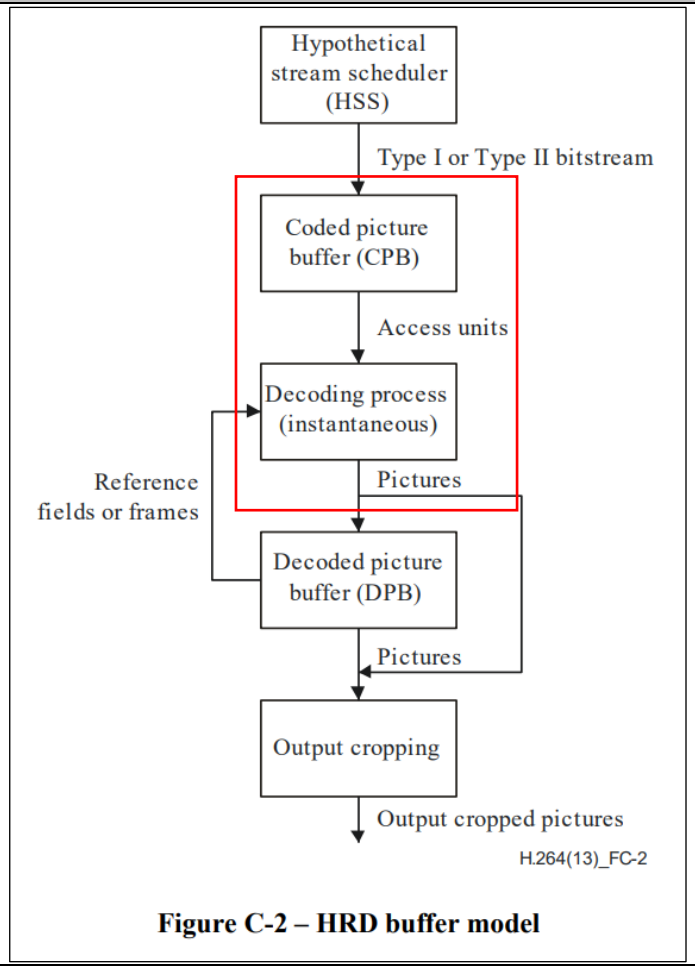
Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	cpb_removal_delay specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2A] decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint;	Users of the Samsung Accused Products perform a method decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A decodes the coded information in the coded multi-view video signal to recover the desired delay times (*e.g.*, *cpb_removal_delay*) for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. The *cpb_removal_delay* value must be decoded in order to determine when to remove the access unit from the CPB.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Annex C **Hypothetical reference decoder** (This annex forms an integral part of this Recommendation | International Standard.) This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2B] decoding the coded signal originating from the picture signal corresponding to	Users of the Samsung Accused Products perform a method decoding the coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the base viewpoint;	For example, the Samsung 8K Neo QLED QN800A decodes the coded signal originating from the picture signal corresponding to the base viewpoint. As the only view that can be independently decoded, and having the minimum index, the base view will be decoded first. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2C] decoding the coded signals originating from the picture signals	Users of the Samsung Accused Products perform a method decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
corresponding to the viewpoints except the base viewpoint; and	For example, the Samsung 8K Neo QLED QN800A decodes the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. For example, other views in the coded sequence will be released from the CPB and instantaneously decoded subsequent to the base view, based on the view index. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2D] delaying the decoding of the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint	Users of the Samsung Accused Products perform a method delaying the decoding of the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint relative to the decoding of the coded signal originating from the picture signal corresponding to the base viewpoint by time intervals equal to the recovered desired delay times. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. With respect to "the recovered desired delay times," the support for Claim [2A] is incorporated by reference.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
relative to the decoding of the coded signal originating from the picture signal corresponding to the base viewpoint by time intervals equal to the recovered desired delay times.	For example, the Samsung 8K Neo QLED QN800A delays the decoding of the coded signals (*e.g.*, other viewpoints) relative to (*e.g.* after) the decoding of the coded signal (*e.g.*, base view) by time intervals equal to the desired delay times (*e.g.*, *cpb_removal_delay*) recovered. The time at which a coded picture for any view (an access unit) is decoded is determined by the CPB removal time. For MVC coded pictures, this removal time is coded in nested Supplemental Enhanced Information (SEI) messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 **Figure C-2 – HRD buffer model**

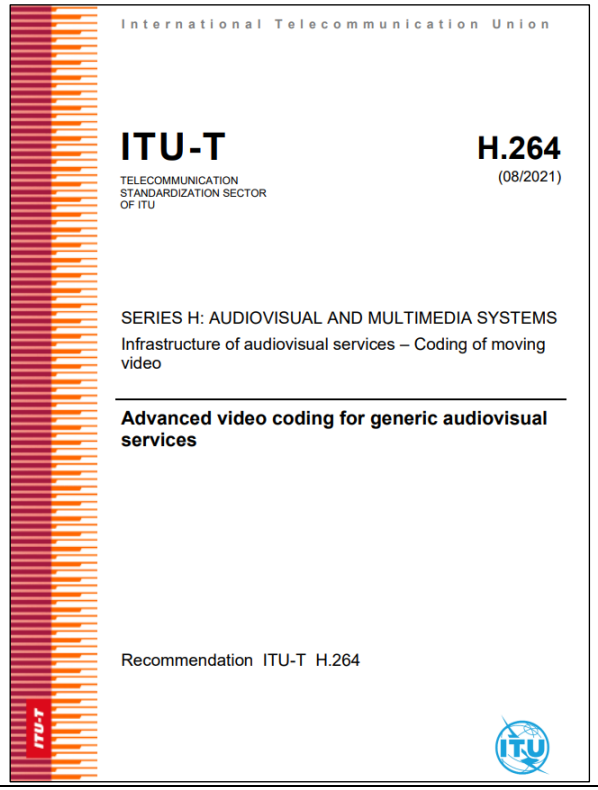
Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3Pa] A non-transitory computer	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding a coded multi-view video signal including coded signals resulting from encoding picture

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
program stored in a computer readable medium for decoding a coded multi-view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints respectively,	signals corresponding to different viewpoints, respectively. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A television supports the H.264 MVC codec.

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf The Samsung 8K Neo QLED QN800A is an apparatus for decoding a coded multi-view video signal (*e.g.*, a coded video sequence conforming to an MVC profile specified in Annex H) including coded signals resulting from encoding picture signals corresponding to different viewpoints (*e.g.*, view components) respectively. H.264 Annex H provides the definition of coding for a multi-view video signal, supporting coding of different viewpoints, including binocular 3D or binocular stereoscopy, as described in the '150 Patent. H.264 Annex C describes the structure and operation of a reference decoder to support the coding standard.  International Telecommunication Union ITU-T TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU H.264 (08/2021) SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving video Advanced video coding for generic audiovisual services Recommendation ITU-T H.264

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The 4th edition of ITU-T H.264, approved in 2009-05, contained enhancement extensions to support multiview video coding (MVC), specification of a "Constrained Baseline Profile", and some miscellaneous corrections and clarifications. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Annex H

Multiview video coding

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies multiview video coding, referred to as MVC.

H.1 Scope

Bitstreams and decoders conforming to the profile specified in this annex are completely specified in this annex with reference made to clauses 2 to 9 and Annexes A to E.

H.3.46 **view:** A sequence of *view components* associated with an identical value of view_id.

H.3.47 **view component:** A *coded representation* of a *view* in a single *access unit*. When profile_idc is equal to 134, a view contains samples of two distinct spatially packed constituent frames that are packed into one frame using one of the frame packing arrangement schemes as specified in clause D.2.26.

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Additionally, the H.264 MVC codec comprises a non-transitory computer program stored in a computer-readable medium because the codec allows motion video to be "manipulated as a form of computer data" and "stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels." **Recommendation ITU-T H.264** **Advanced video coding for generic audiovisual services** **Summary** Recommendation ITU-T H.264 | International Standard ISO/IEC 14496-10 represents an evolution of the existing video coding standards (ITU-T H.261, ITU-T H.262, and ITU-T H.263) and it was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, Internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3Pb] wherein at least one among the viewpoints is designated as a base viewpoint,	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding a coded multi-view video signal wherein at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint. Samsung directly

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint, and	infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, in multi-view decoding, at least one among the viewpoints is designated as a base viewpoint (*e.g.*, base view), the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints (*e.g.*, independently of other views) except the base viewpoint. For example, the base view can be decoded independently (without referring to) any other view and there is only one base view in a coded video sequence. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3Pc] wherein the coded multi-view video signal contains coded information representing a desired delay	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding a coded multi-view video signal wherein the coded multi-view video signal contains coded information representing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC).

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint, the computer program comprising the steps of:	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, the coded multi-view video signal (*e.g.*, MVC coded video sequence) contains coded information representing a desired delay time (*e.g.*, *cpb_removal_delay*, picture timing SEI messages that are included in MVC scalable nesting Supplemental Enhanced Information (SEI)messages) of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. The time at which a coded picture for any view (an access unit) is decoded is determined by the coded picture buffer (CPB) removal time. For MVC coded pictures, this removal time is coded in nested SEI messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

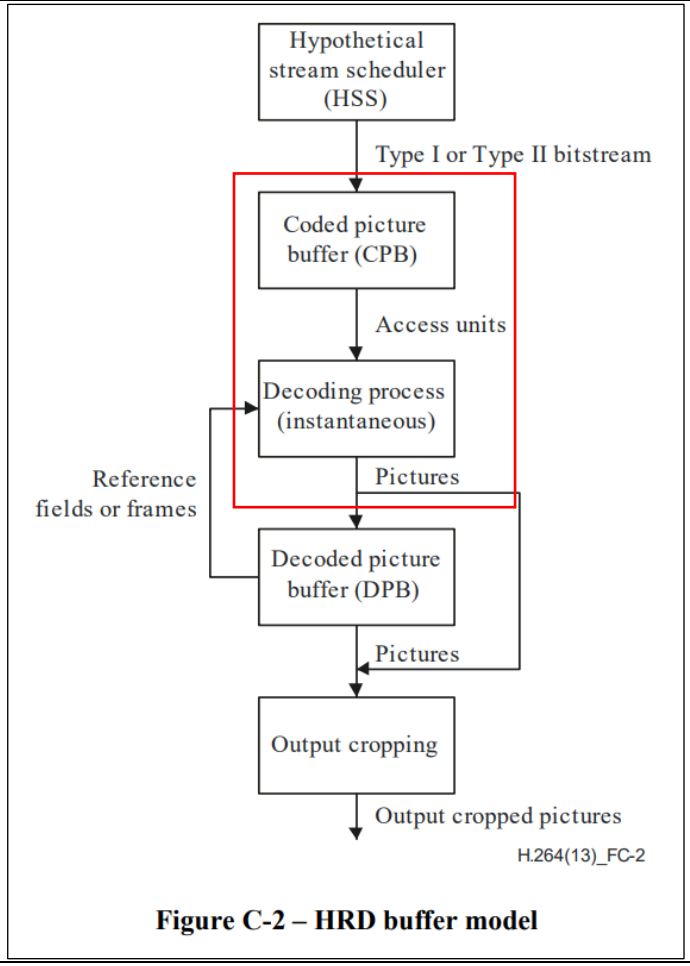


Figure C-2 – HRD buffer model

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (a) When an access unit contains one or more buffering period SEI messages that are included in MVC scalable nesting SEI messages, the buffering period SEI message that is associated with the operation point being decoded is the buffering period SEI message that initializes the HRD. Let hrdVId[i] be equal to sei_op_view_id[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and let hrdTId be the value of sei_op_temporal_id, that are associated with the MVC scalable nesting SEI message containing the buffering period SEI message that initializes the HRD. (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. (c) The HRD parameters that are used for conformance checking are the HRD parameters included in the MVC video usability information extension of the active MVC sequence parameter set that are associated with values of vui_mvc_view_id[i][j] for all j in the range of 0 to vui_mvc_num_target_output_views_minus1[i], inclusive, equal to hrdVId[j], and the value of vui_mvc_temporal_id[i] equal to hrdTId. For the specification in this annex, num_units_in_tick, time_scale, fixed_frame_rate_flag, nal_hrd_parameters_present_flag, vcl_hrd_parameters_present_flag, low_delay_hrd_flag, and pic_struct_present_flag are substituted with the values of vui_mvc_num_units_in_tick[i], vui_mvc_time_scale[i], vui_mvc_fixed_frame_rate_flag[i], vui_mvc_nal_hrd_parameters_present_flag[i], vui_mvc_vcl_hrd_parameters_present_flag[i], vui_mvc_low_delay_hrd_flag[i], and vui_mvc_pic_struct_present_flag[i], respectively, with i being the value for which vui_mvc_view_id[i] is equal to hrdVId[j] for all j in the range of 0 to vui_mvc_num_traget_output_views_minus1[i], inclusive, and vui_mvc_temporal_id[i] equal to hrdTId.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	cpb_removal_delay specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3A] decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint;	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding the coded information in the coded multi-view video signal to recover the desired delay times for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program for decoding the coded information in the coded multi-view video signal to recover the desired delay times (*e.g.*, *cpb_removal_delay*) for the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. The *cpb_removal_delay* value must be decoded in order to determine when to remove the access unit from the CPB.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Annex C **Hypothetical reference decoder** (This annex forms an integral part of this Recommendation | International Standard.) This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3B] decoding the coded signal originating from the picture signal corresponding to	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding the coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the base viewpoint;	For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program for decoding the coded signal originating from the picture signal corresponding to the base viewpoint. As the only view that can be independently decoded, and having the minimum index, the base view will be decoded first. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3C] decoding the coded signals originating from the picture	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC).

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
signals corresponding to the viewpoints except the base viewpoint; and	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program for decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. For example, other views in the coded sequence will be released from the CPB and instantaneously decoded subsequent to the base view, based on the view index. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3D] delaying the decoding of the coded signals originating from the picture signals corresponding to	The Samsung Accused Products include a non-transitory computer program stored in a computer readable medium for delaying the decoding of the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint relative to the decoding of the coded signal originating from the picture signal corresponding to the base viewpoint by time intervals equal to the recovered desired delay times. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices support H.264 Multi View Coding (MVC). This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
the viewpoints except the base viewpoint relative to the decoding of the coded signal originating from the picture signal corresponding to the base viewpoint by time intervals equal to the recovered desired delay times.	With respect to "the recovered desired delay times," the support for Claim [3A] is incorporated by reference. For example, the Samsung 8K Neo QLED QN800A uses a non-transitory computer program for delaying the decoding of the coded signals (*e.g.*, other viewpoints) relative to (*e.g.* after) the decoding of the coded signal (*e.g.*, base view) by time intervals equal to the desired delay times (*e.g.*, *cpb_removal_delay*) recovered. The time at which a coded picture for any view (an access unit) is decoded is determined by the CPB removal time. For MVC coded pictures, this removal time is coded in nested Supplemental Enhanced Information (SEI) messages. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

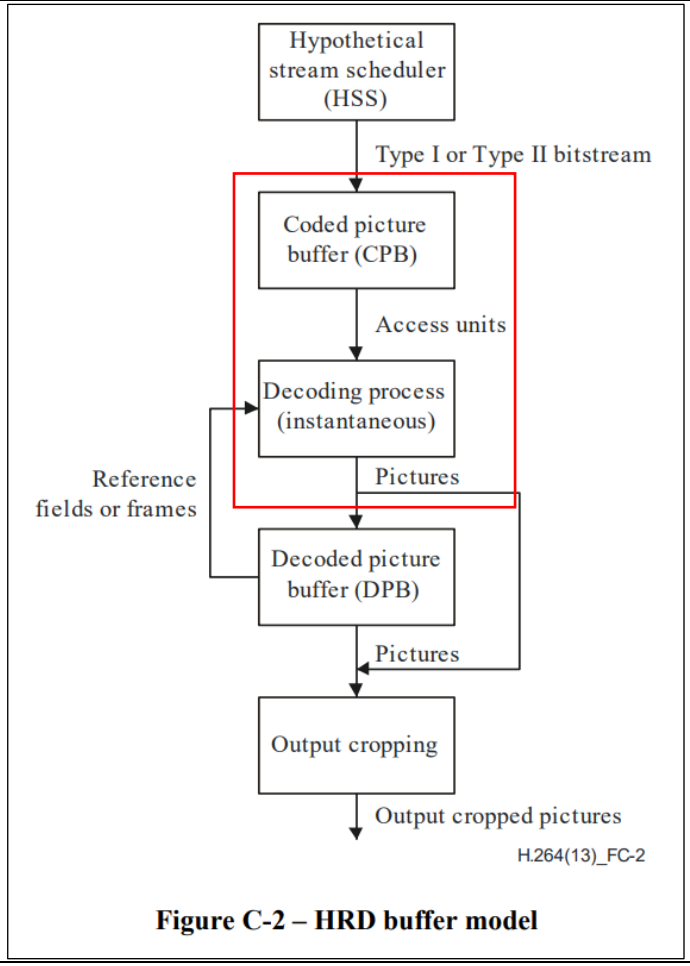


Figure C-2 – HRD buffer model

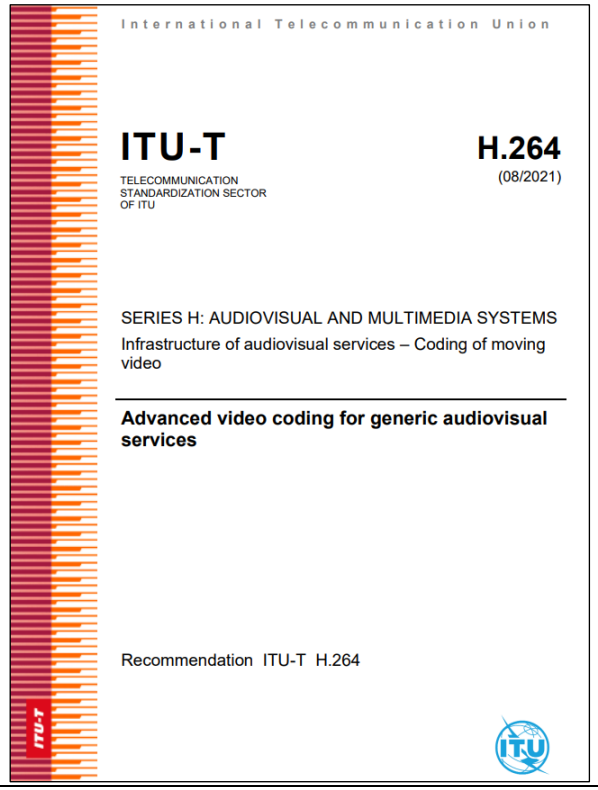
Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4Pa] A method of decoding a coded multi-	Users of the Samsung Accused Products perform a method of decoding a coded multi-view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints, respectively.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
view video signal including coded signals resulting from encoding picture signals corresponding to different viewpoints respectively,	Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A television supports the H.264 MVC codec.

Supported video codecs (QN7A Series or higher)**

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
*.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The Samsung 8K Neo QLED QN800A decodes a coded multi-view video signal (*e.g.*, a coded video sequence conforming to an MVC profile specified in Annex H) including coded signals resulting from encoding picture signals corresponding to different viewpoints (*e.g.*, view components) respectively. H.264 Annex H provides the definition of coding for a multi-view video signal, supporting coding of different viewpoints, including binocular 3D or binocular stereoscopy, as described in the '150 Patent. H.264 Annex C describes the structure and operation of a reference decoder to support the coding standard.  International Telecommunication Union ITU-T TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU H.264 (08/2021) SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving video **Advanced video coding for generic audiovisual services** Recommendation ITU-T H.264

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The 4th edition of ITU-T H.264, approved in 2009-05, contained enhancement extensions to support multiview video coding (MVC), specification of a "Constrained Baseline Profile", and some miscellaneous corrections and clarifications. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Annex H

Multiview video coding

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies multiview video coding, referred to as MVC.

H.1 Scope

Bitstreams and decoders conforming to the profile specified in this annex are completely specified in this annex with reference made to clauses 2 to 9 and Annexes A to E.

H.3.46 **view:** A sequence of *view components* associated with an identical value of view_id.

H.3.47 **view component:** A *coded representation* of a *view* in a single *access unit*. When profile_idc is equal to 134, a view contains samples of two distinct spatially packed constituent frames that are packed into one frame using one of the frame packing arrangement schemes as specified in clause D.2.26.

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[4Pb] wherein at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint, and	In the Samsung Accused Products, at least one among the viewpoints is designated as a base viewpoint, the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints except the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, in multi-view decoding, at least one among the viewpoints is designated as a base viewpoint (*e.g.*, base view), the base viewpoint corresponding to a picture signal which is encoded without referring to at least one picture signal corresponding to one of the viewpoints (*e.g.*, independently of other views) except the base viewpoint. For example, the base view can be decoded independently (without referring to) any other view and there is only one base view in a coded video sequence. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4Pc] wherein the coded multi-view video signal contains coded information	In the Samsung Accused Products, the coded multi-view video signal contains coded information about view dependencies representing which of the picture signals are referred to as reference signals for the encoding of each of the picture signals. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
about view dependencies representing which of the picture signals are referred to as reference signals for the encoding of each of the picture signals, the method comprising the steps of:	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, the coded multi-view video signal (*e.g.*, MVC coded video sequence) contains coded information about view dependencies representing which of the picture signals are referred to as reference signals for the encoding of each of the picture signals (i.e., reference pictures). Specifically, to determine whether a picture signal is referred to as a reference signal, a coded picture has a "nal_ref_idc syntax element that is associated with the target dependency representation" (*i.e.*, coded information about view dependencies) for the purpose of inter prediction in the decoding process of subsequent non-reference signals in the decoding order. **G.3.43 reference base field**: A *reference field* that is obtained by decoding a *base quality layer representation* with the nal_ref_idc *syntax element* not equal to 0, the store_ref_base_pic_flag *syntax element* equal to 1, and the field_pic_flag *syntax element* equal to 1 of a *coded picture* and all *layer representations* of the *coded picture* that are referred to by *inter-layer prediction* in the *base quality layer representation*. A reference base field is not a *decoded picture* and it is not an output of the *decoding process*, but may be used for *inter prediction* when *P*, *B*, *EP*, and *EB slices* of a *coded field* or a *field macroblock* of a *coded frame* are decoded. See also *reference base picture*. **G.3.44 reference base frame**: A *reference frame* that is obtained by decoding a *base quality layer representation* with the nal_ref_idc *syntax element* not equal to 0, the store_ref_base_pic_flag *syntax element* equal to 1, and the field_pic_flag *syntax element* equal to 0 of a *coded picture* and all *layer representations* of the *coded picture* that are referred to by *inter-layer prediction* of the *base quality layer representation*. A reference base frame is not a *decoded picture* and it is not an output of the *decoding process*, but may be used for *inter prediction* when *P*, *B*, *EP*, and *EB slices* of a *coded frame* are decoded. See also *reference base picture*. **G.3.45 reference base picture**: A collective term for a *reference base field* or a *reference base frame*.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

G.3.46 **reference field**: A *reference field* may be used for *inter prediction* when *P*, *B*, *EP*, or *EB slices* of a *coded field* or *field macroblocks* of a *coded frame* are decoded. See also *reference picture*.

G.3.47 **reference frame**: A *reference frame* may be used for *inter prediction* when *P*, *B*, *EP*, or *EB slices* of a *coded frame* are decoded. See also *reference picture*.

G.3.48 **reference layer macroblock**: A *macroblock* of a *reference layer representation*.

G.3.49 **reference layer representation**: A reference layer representation for a particular *layer representation* of a *coded picture* is the *layer representation* that is used for *inter-layer prediction* of the particular *layer representation*. The reference layer representation belongs to the same *access unit* as the *layer representation* that uses the reference layer representation for *inter-layer prediction*.

G.3.50 **reference picture**: A collective term for a *decoded picture* that is obtained by decoding a *coded picture* for which the nal_ref_idc *syntax element* that is associated with the *target dependency representation* is not equal to 0 or a *reference base picture*. A reference picture contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.

G.3.51 **reference picture list**: A list of *reference pictures* that is used for *inter prediction* of a *P*, *B*, *EP*, or *EB slice*. For the *decoding process* of a *P* or *EP slice*, there is one reference picture list. For the *decoding process* of a *B* or *EB slice*, there are two reference picture lists.

G.3.52 **reference picture list 0**: A *reference picture list* used for *inter prediction* of a *P*, *B*, *EP*, or *EB slice*. All *inter prediction* used for *P* and *EP slices* uses reference picture list 0. Reference picture list 0 is one of two *reference picture lists* used for *inter prediction* for a *B* or *EB slice*, with the other being *reference picture list 1*.

G.3.53 **reference picture list 1**: A *reference picture list* used for *inter prediction* of a *B* or *EB slice*. Reference picture list 1 is one of two *reference picture lists* used for *inter prediction* for a *B* or *EB slice*, with the other being *reference picture list 0*.

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

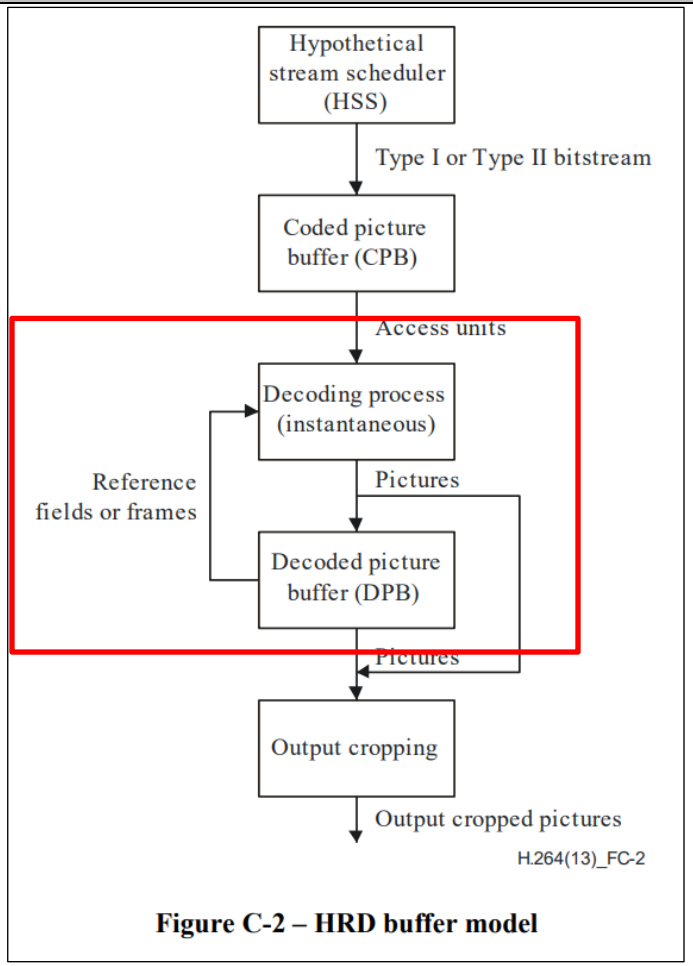


Figure C-2 – HRD buffer model

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4A] decoding the coded information in the coded multi-view video signal to recover the view dependencies;	Users of the Samsung Accused Products perform decoding the coded information in the coded multi-view video signal to recover the view dependencies. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A decodes the coded information in the coded multi-view video signal to recover view dependencies indicating the reference pictures for interframe prediction are fed back into the "decoding process (instantaneous)" for decoding so that the view dependencies can be recovered by the decoder.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Annex C **Hypothetical reference decoder** (This annex forms an integral part of this Recommendation | International Standard.) This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

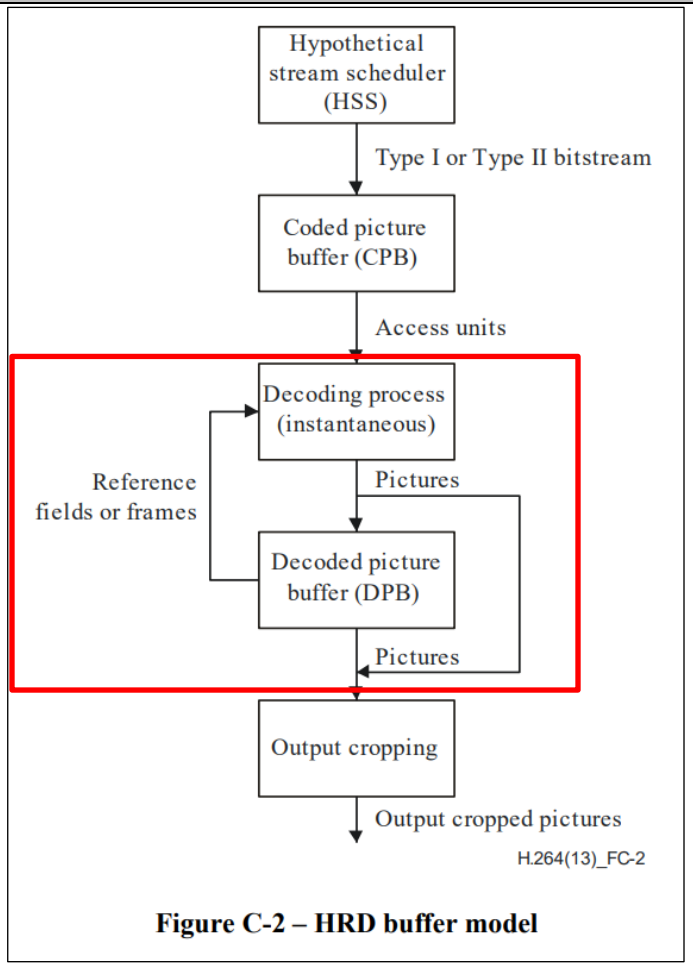
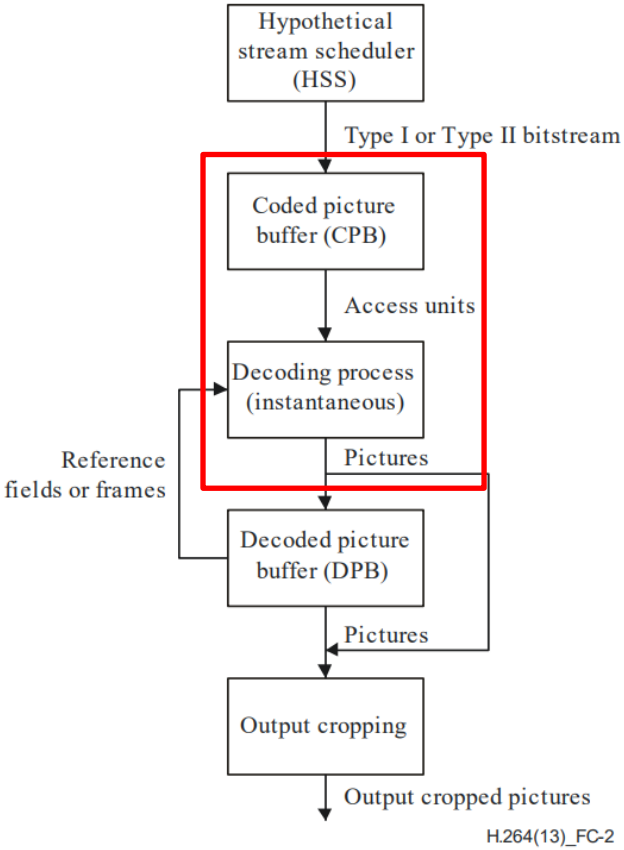


Figure C-2 – HRD buffer model

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items
[4B] computing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base	Users of the Samsung Accused Products perform computing a desired delay time of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint in response to the recovered view dependencies. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in Samsung 8K Neo QLED QN800A, the coded multi-view video signal (*e.g.*, MVC coded video sequence) contains coded information, which can be used to compute a desired delay time (*e.g.*, *cpb_removal_delay*, picture timing SEI messages that are included in MVC scalable nesting Supplemental Enhanced Information (SEI)messages) of the start of decoding a coded signal originating from each of the picture signals corresponding to the viewpoints except the base viewpoint relative to the start of decoding a coded signal originating from the picture signal corresponding to the base viewpoint. The time at which a coded picture for any view (an access unit) is decoded is determined by the coded picture buffer (CPB) removal time. For MVC coded pictures, this removal time is coded in nested SEI messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
viewpoint in response to the recovered view dependencies;	The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.  **Figure C-2 – HRD buffer model**

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (a) When an access unit contains one or more buffering period SEI messages that are included in MVC scalable nesting SEI messages, the buffering period SEI message that is associated with the operation point being decoded is the buffering period SEI message that initializes the HRD. Let hrdVId[i] be equal to sei_op_view_id[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and let hrdTId be the value of sei_op_temporal_id, that are associated with the MVC scalable nesting SEI message containing the buffering period SEI message that initializes the HRD. (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. (c) The HRD parameters that are used for conformance checking are the HRD parameters included in the MVC video usability information extension of the active MVC sequence parameter set that are associated with values of vui_mvc_view_id[i][j] for all j in the range of 0 to vui_mvc_num_target_output_views_minus1[i], inclusive, equal to hrdVId[j], and the value of vui_mvc_temporal_id[i] equal to hrdTId. For the specification in this annex, num_units_in_tick, time_scale, fixed_frame_rate_flag, nal_hrd_parameters_present_flag, vcl_hrd_parameters_present_flag, low_delay_hrd_flag, and pic_struct_present_flag are substituted with the values of vui_mvc_num_units_in_tick[i], vui_mvc_time_scale[i], vui_mvc_fixed_frame_rate_flag[i], vui_mvc_nal_hrd_parameters_present_flag[i], vui_mvc_vcl_hrd_parameters_present_flag[i], vui_mvc_low_delay_hrd_flag[i], and vui_mvc_pic_struct_present_flag[i], respectively, with i being the value for which vui_mvc_view_id[i] is equal to hrdVId[j] for all j in the range of 0 to vui_mvc_num_traget_output_views_minus1[i], inclusive, and vui_mvc_temporal_id[i] equal to hrdTId.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	cpb_removal_delay specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items More specifically, computing the desired delay time (i.e., *cpb_removal_delay*) is in response to the recovered view dependencies from element [4A] because the parameter *cpb_removal_delay* indicates the delay relative to the base viewpoints indicated by the recovered view dependencies. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

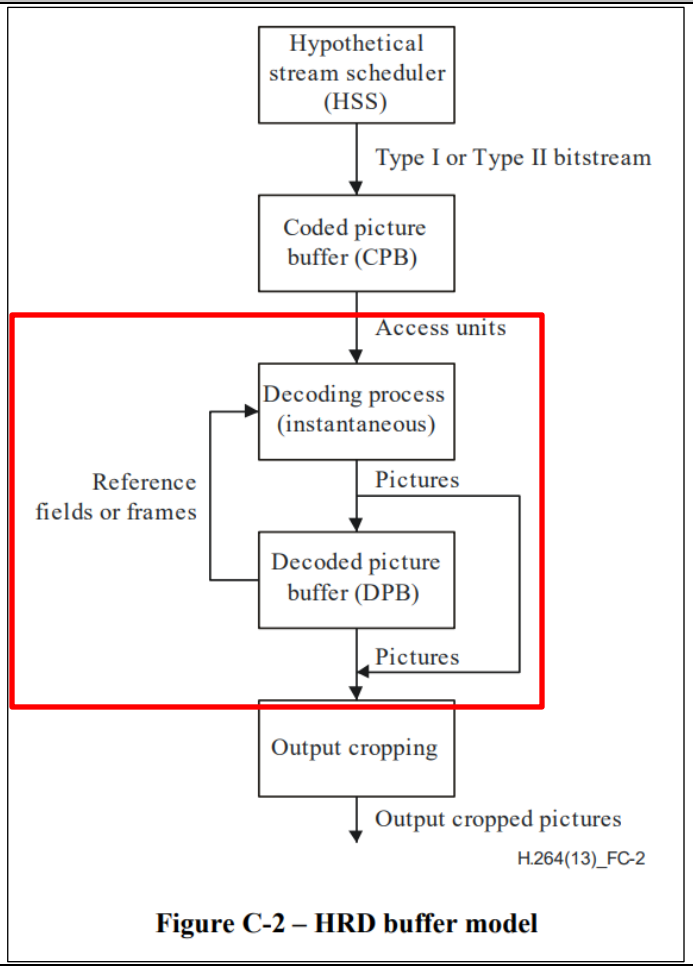


Figure C-2 – HRD buffer model

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference".
	When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (a) When an access unit contains one or more buffering period SEI messages that are included in MVC scalable nesting SEI messages, the buffering period SEI message that is associated with the operation point being decoded is the buffering period SEI message that initializes the HRD. Let hrdVId[i] be equal to sei_op_view_id[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and let hrdTId be the value of sei_op_temporal_id, that are associated with the MVC scalable nesting SEI message containing the buffering period SEI message that initializes the HRD. (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. (c) The HRD parameters that are used for conformance checking are the HRD parameters included in the MVC video usability information extension of the active MVC sequence parameter set that are associated with values of vui_mvc_view_id[i][j] for all j in the range of 0 to vui_mvc_num_target_output_views_minus1[i], inclusive, equal to hrdVId[j], and the value of vui_mvc_temporal_id[i] equal to hrdTId. For the specification in this annex, num_units_in_tick, time_scale, fixed_frame_rate_flag, nal_hrd_parameters_present_flag, vcl_hrd_parameters_present_flag, low_delay_hrd_flag, and pic_struct_present_flag are substituted with the values of vui_mvc_num_units_in_tick[i], vui_mvc_time_scale[i], vui_mvc_fixed_frame_rate_flag[i], vui_mvc_nal_hrd_parameters_present_flag[i], vui_mvc_vcl_hrd_parameters_present_flag[i], vui_mvc_low_delay_hrd_flag[i], and vui_mvc_pic_struct_present_flag[i], respectively, with i being the value for which vui_mvc_view_id[i] is equal to hrdVId[j] for all j in the range of 0 to vui_mvc_num_traget_output_views_minus1[i], inclusive, and vui_mvc_temporal_id[i] equal to hrdTId.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	cpb_removal_delay specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4C] decoding the coded signal originating from the picture signal corresponding to the base viewpoint;	Users of the Samsung Accused Products perform decoding the coded signal originating from the picture signal corresponding to the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A decodes the coded signal originating from the picture signal corresponding to the base viewpoint. As the only view that can be independently decoded, and having the minimum index, the base view will be decoded first. **H.3.6 base view**: A *view* that has the minimum value of *view order index* in a *coded video sequence*. The base view can be decoded independently of other *views*, does not use *inter-view prediction*, and contains *VCL NAL units* only with nal_unit_type equal to 1, 5, or 14. The *bitstream subset* corresponding to the base view conforms to one or more of the *profiles* specified in Annex A. There is only one base view in a *coded video sequence*.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4D] decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint; and	Users of the Samsung Accused Products perform decoding the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A decodes the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint. For example, other views in the coded sequence will be released from the CPB and instantaneously decoded subsequent to the base view, based on the view index. When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified:

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	(b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4E] delaying the decoding of the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint relative to the decoding of the coded signal originating from the picture signal corresponding to the base	Users of the Samsung Accused Products perform delaying the decoding of the coded signals originating from the picture signals corresponding to the viewpoints except the base viewpoint relative to the decoding of the coded signal originating from the picture signal corresponding to the base viewpoint by time intervals equal to the computed desired delay times. Samsung directly infringes and induces infringement by causing users to perform the claimed method on devices having the H.264 MVC codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. With respect to "computed desired delay times," the support for Claim [4B] is incorporated by reference. For example, the Samsung 8K Neo QLED QN800A delays the decoding of the coded signals (*e.g.*, other viewpoints) relative to (*e.g.* after) the decoding of the coded signal (*e.g.*, base view) by time intervals equal to the computed desired delay times (*e.g.*, *cpb_removal_delay*). The time at which a coded picture for any view (an access unit) is decoded is determined by the CPB removal time. For MVC coded pictures, this removal time is coded in nested Supplemental Enhanced Information (SEI) messages, defined in H.264 section D. Specifically, the parameter *cpb_removal_delay* determines the delay time for removing the coded data from the CPB relative to decoding of the SEI messages. Since the base view is the only view that can be independently decoded, it must

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
viewpoint by time intervals equal to the computed desired delay times.	be decoded before other views, so the *cpb_removal_delay* messages for other views will then be relative to the base view. The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products

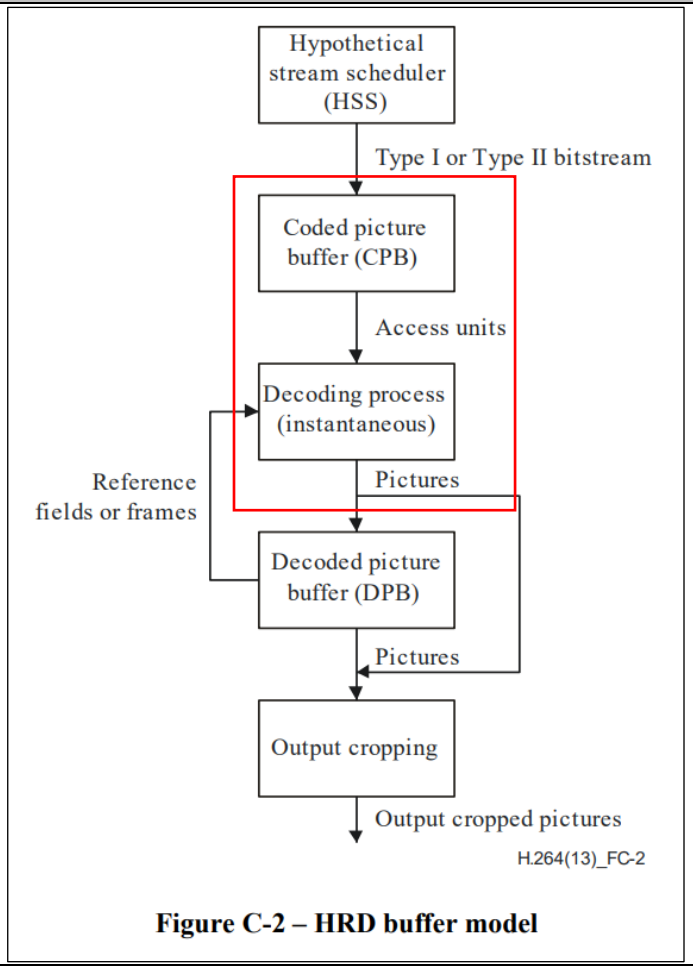


Figure C-2 – HRD buffer model

Claim – 8,139,150	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference". When the coded video sequence conforms to one or more of the profiles specified in Annex H and the decoding process specified in Annex H is applied, the following is specified: (b) The picture timing SEI messages that specify the removal timing of access units from the CPB and output timing from the DPB are the picture timing SEI messages that are included in MVC scalable nesting SEI messages associated with values of sei_op_view_id[i] equal to hrdVId[i] for all i in the range of 0 to num_view_components_op_minus1, inclusive, and sei_temporal_id equal to hrdTId. **cpb_removal_delay** specifies how many clock ticks (see clause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message in a preceding access unit before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code having a length in bits given by cpb_removal_delay_length_minus1 + 1. The cpb_removal_delay is the remainder of a modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$ counter. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-202108-I!!PDF-E&type=items Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claims 1, 2, 4, and 6 (the "Asserted Claims") of U.S. Patent No. 9,445,041 (the "'041 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung Galaxy S22 Ultra smartphone, depicted in this chart, is believed to be representative of other smartphones and tablets including at least the Galaxy A13 5G, Galaxy A22 5G, Galaxy A32 5G, Galaxy A33 5G, Galaxy A40, Galaxy A42 5G, Galaxy A50, Galaxy A50s, Galaxy A51 5G, Galaxy A52 5G, Galaxy A52s 5G, Galaxy A53 5G, Galaxy A70, Galaxy A70s, Galaxy A71 5G, Galaxy A73 5G, Galaxy A80, Galaxy A90 5G, Galaxy S9, Galaxy S9+, Galaxy Note9, Galaxy S10e, Galaxy S10, Galaxy S10+, Galaxy S10 Lite, Galaxy Fold 5G, Galaxy Note 10 5G, Galaxy Note 10+ 5G, Galaxy Note 20 5G, Galaxy Note 20 Ultra 5G, Galaxy Quantum 2, Galaxy S10 5G, Galaxy S20 5G, Galaxy S20 FE 5G, Galaxy S20 Ultra 5G, Galaxy S20 UW, Galaxy S20+ 5G, Galaxy S21 5G, Galaxy S21 Ultra 5G, Galaxy S21+ 5G, Galaxy S22, Galaxy S22 5G, Galaxy S22 Ultra, Galaxy S22+, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra, Galaxy Tab S7 5G, Galaxy Tab S7 FE 5G, Galaxy Tab S7+ 5G, Galaxy Tab S8+ 5G, Galaxy Z Flip 5G, Galaxy Z Flip3 5G, Galaxy Z Flip4 5G, Galaxy Z Fold 2 5G, the Galaxy Z Fold3 5G, Galaxy Z Fold4 5G models for the purposes of infringement of the '041 Patent. Based on publicly available information, ACT believes that the Samsung Galaxy S22 Ultra, charted herein, and other smartphones and tablets share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the smartphones and tablets all have the Samsung Video Editor installed which includes all phones and tablets with Samsung UI 4.1 or higher. Should discovery yield documents or information demonstrating that Defendants sell other infringing products which are not represented by the Samsung Galaxy S22 Ultra smartphone, ACT reserves its right to amend its infringement contentions to include a chart mapping the claims of the '041 Patent to the structure and function of the newly-discovered products. Samsung directly infringes each of the Asserted Claims by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

Samsung indirectly infringes the Asserted Claims in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through their operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claims; and

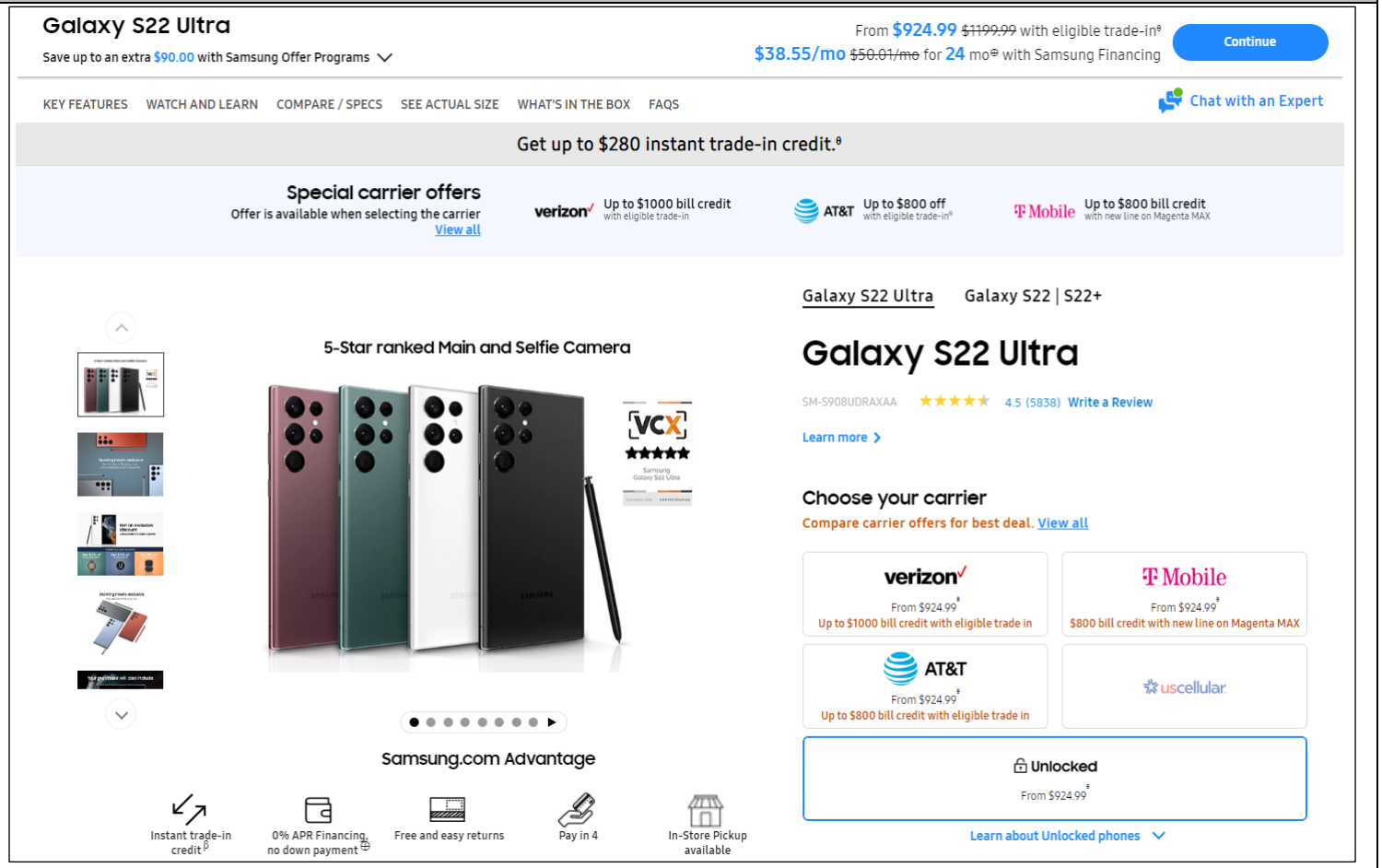
(ii) marketing and advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers, including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claims, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claims but has taken deliberate actions to avoid learning these facts.

Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

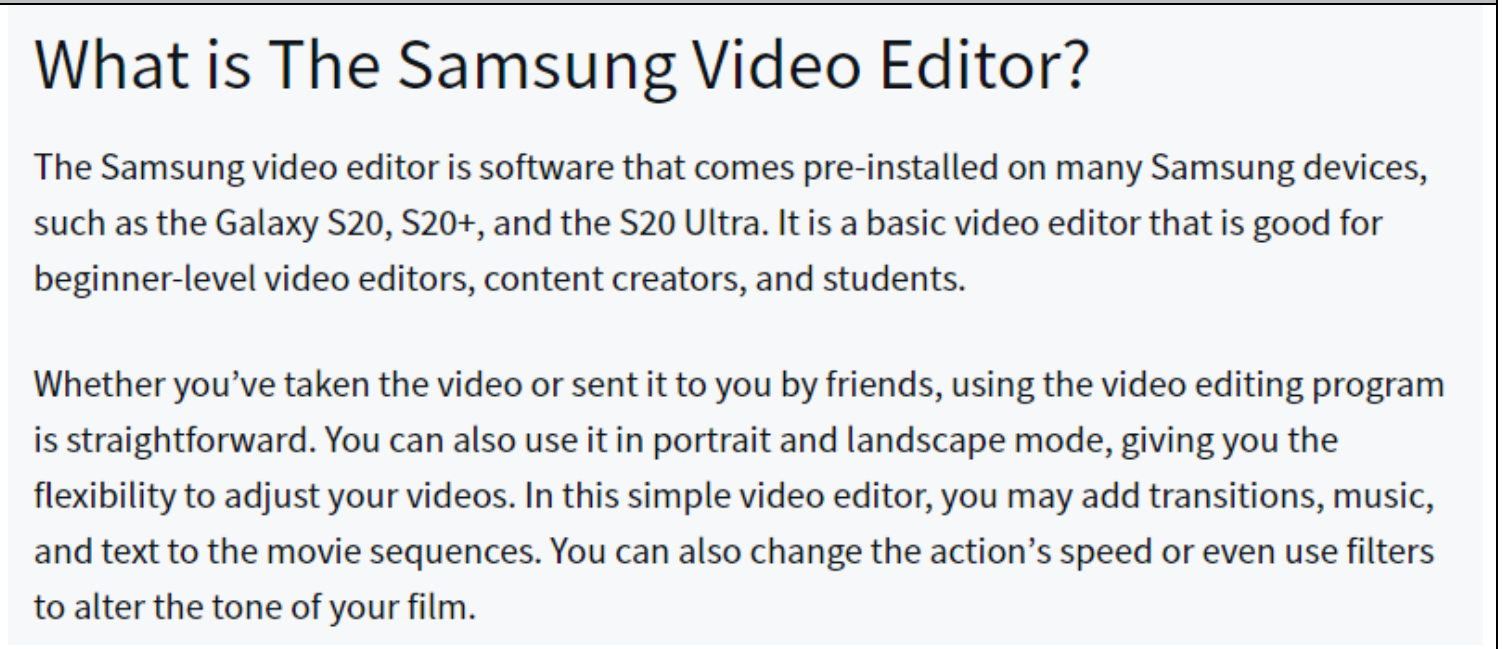
ACT does not concede that any claims of the '041 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

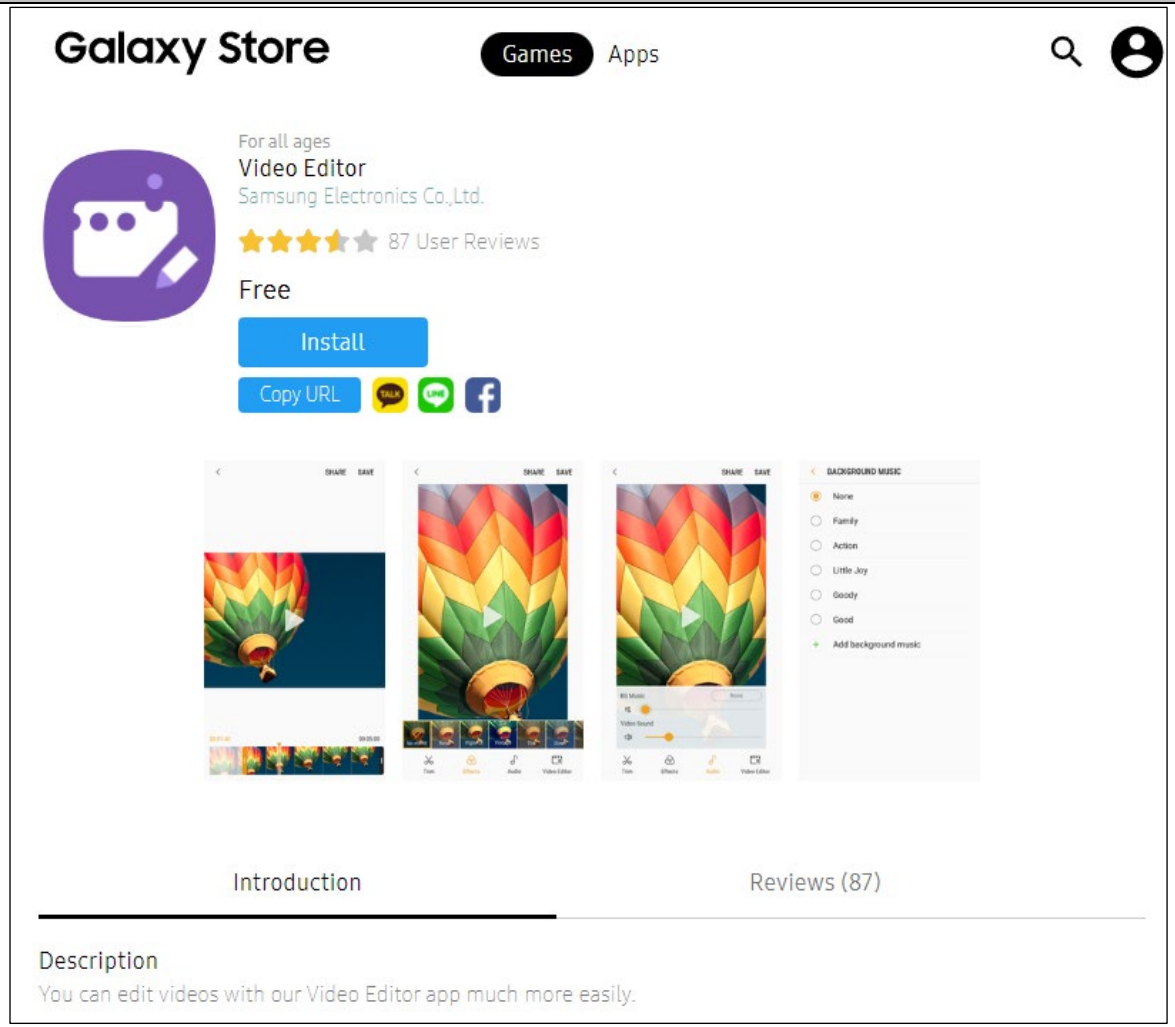
The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

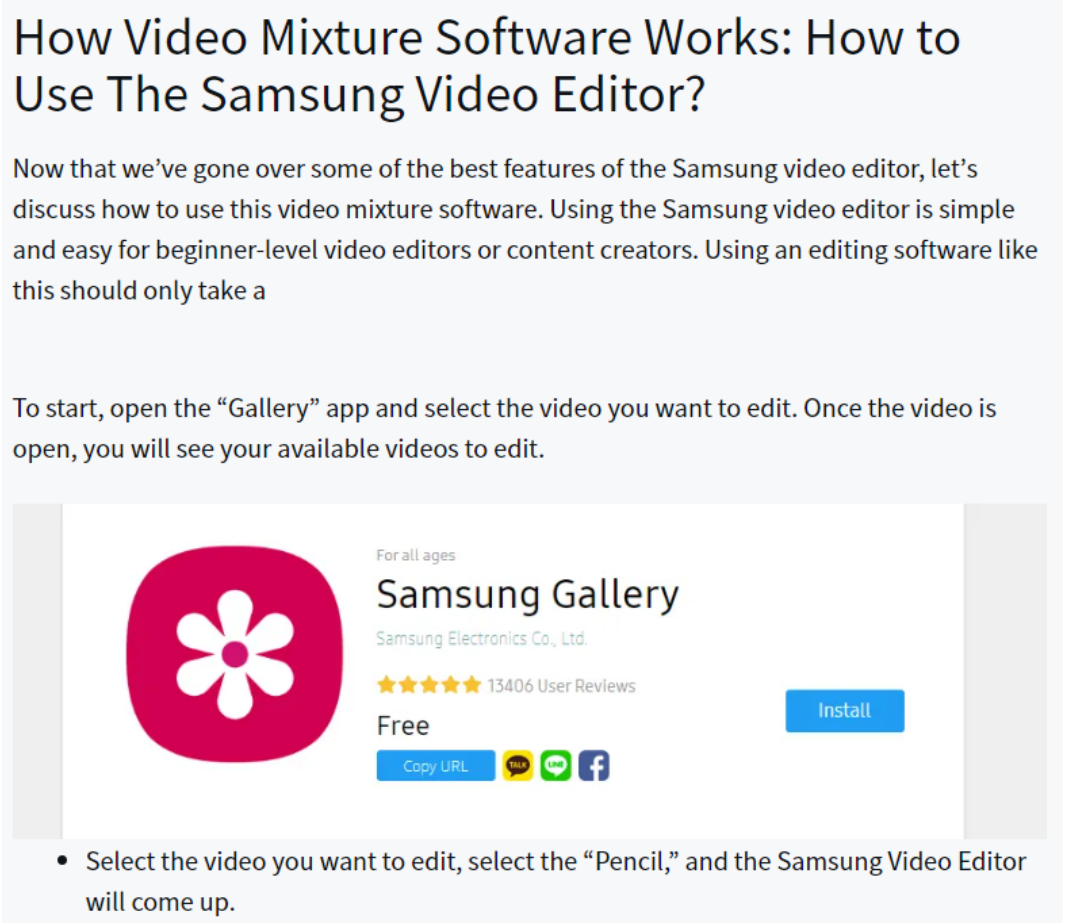
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1P] A moving image data processing apparatus comprising:	The Samsung Accused Products comprise: A moving image data processing apparatus. Samsung infringes directly, and/or indirectly by providing devices such as smartphones and tablets with the Samsung Video Editor installed that are each a moving image data processing apparatus, and encouraging others, including customers and end users, to download the Samsung Video Editor app onto their hardware devices. Samsung also infringes directly or indirectly by providing specific Samsung software updates including the Samsung Video Editor app onto hardware devices of their customers and end users. Samsung directly infringes by using hardware devices with the Samsung Video Editor app installed, such as during testing and qualification. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone with Samsung Gallery and the Samsung Video Editor is an apparatus that processes moving image data, such as videos. On information and belief, Samsung itself tests and uses smartphones and tablets having the Samsung Gallery and Samsung Video Editor app installed. Additionally, Samsung provides the Samsung Gallery and Samsung Video Editor app to customers and users on its smartphones and devices and in its app store.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	

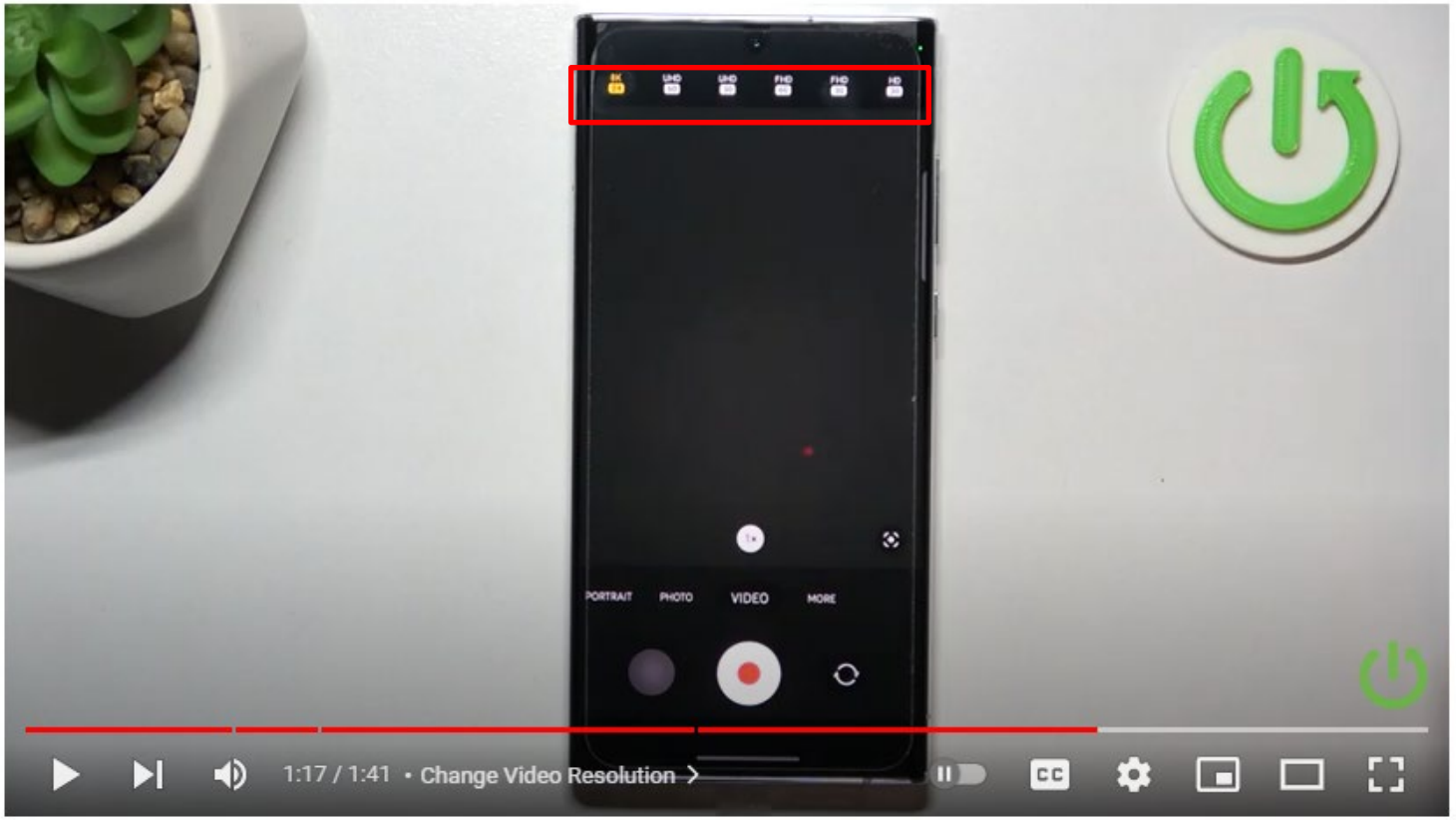
https://www.samsung.com/us/smartphones/galaxy-s22-ultra/buy/galaxy-s22-ultra-128gb-unlocked-sm-s908udraxaa/?modelCode=SM-S908UZAAXAA

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://galaxystore.samsung.com/prepost/000004444189?langCd=en

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1A] a moving image data input unit configured to be input of first moving image data generated at a first frame rate and second image data generated at a second frame rate that is different from the first frame rate;	The Samsung Accused Products comprise: A moving image data input unit configured to be input of first moving image data generated at a first frame rate and second image data generated at a second frame rate that is different from the first frame rate. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone's video camera is a moving image data input unit configured to be input of first moving image data generated at a first frame rate (*e.g.*, Video 1 at 30 frames per second) and second image data generated at a second frame rate that is different from the first frame rate (*e.g.*, Video 2 at 60 frames per second).

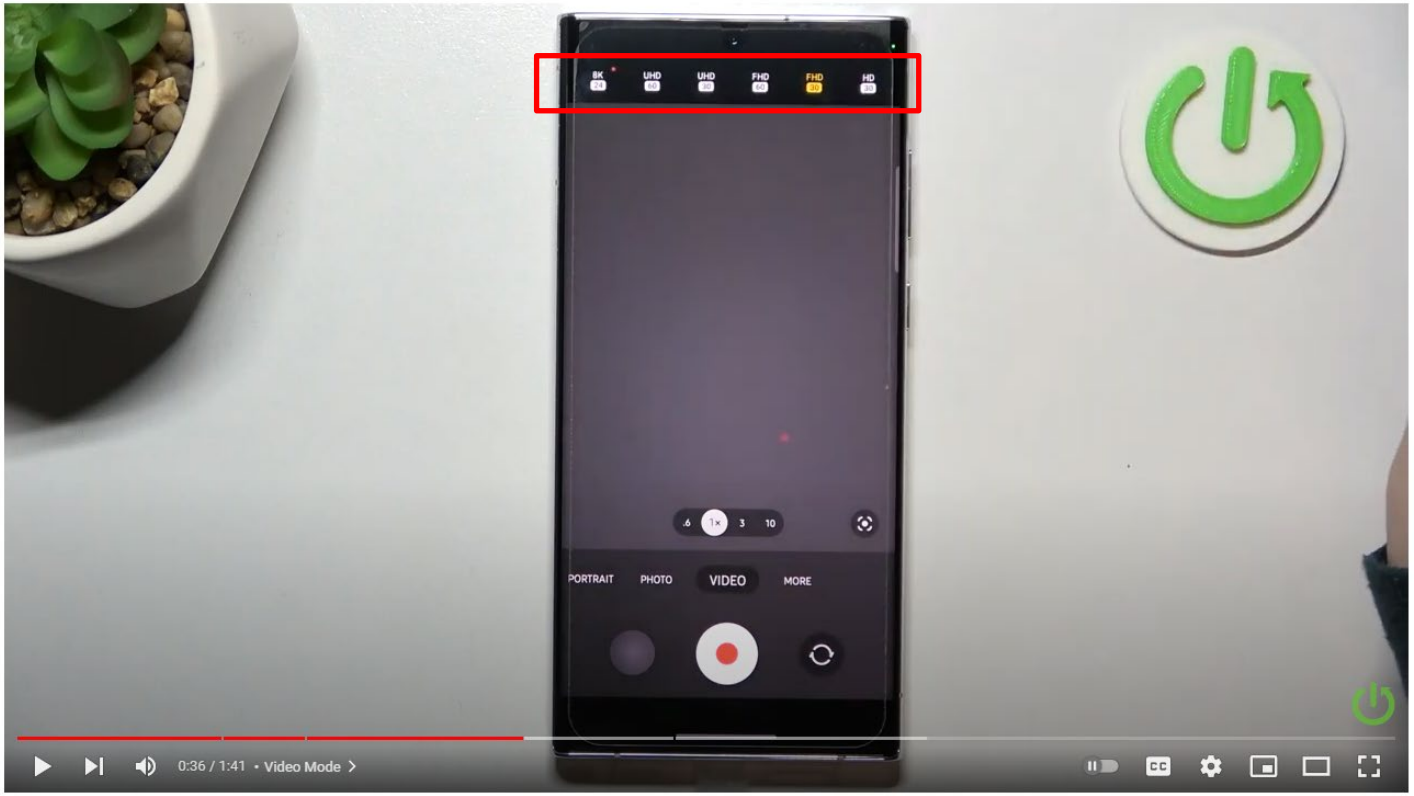
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 How to Change Video Resultion on SAMSUNG Galaxy S22 Ultra - Set Up Video Resolution https://www.youtube.com/watch?v=oL1JQ4uzFDg

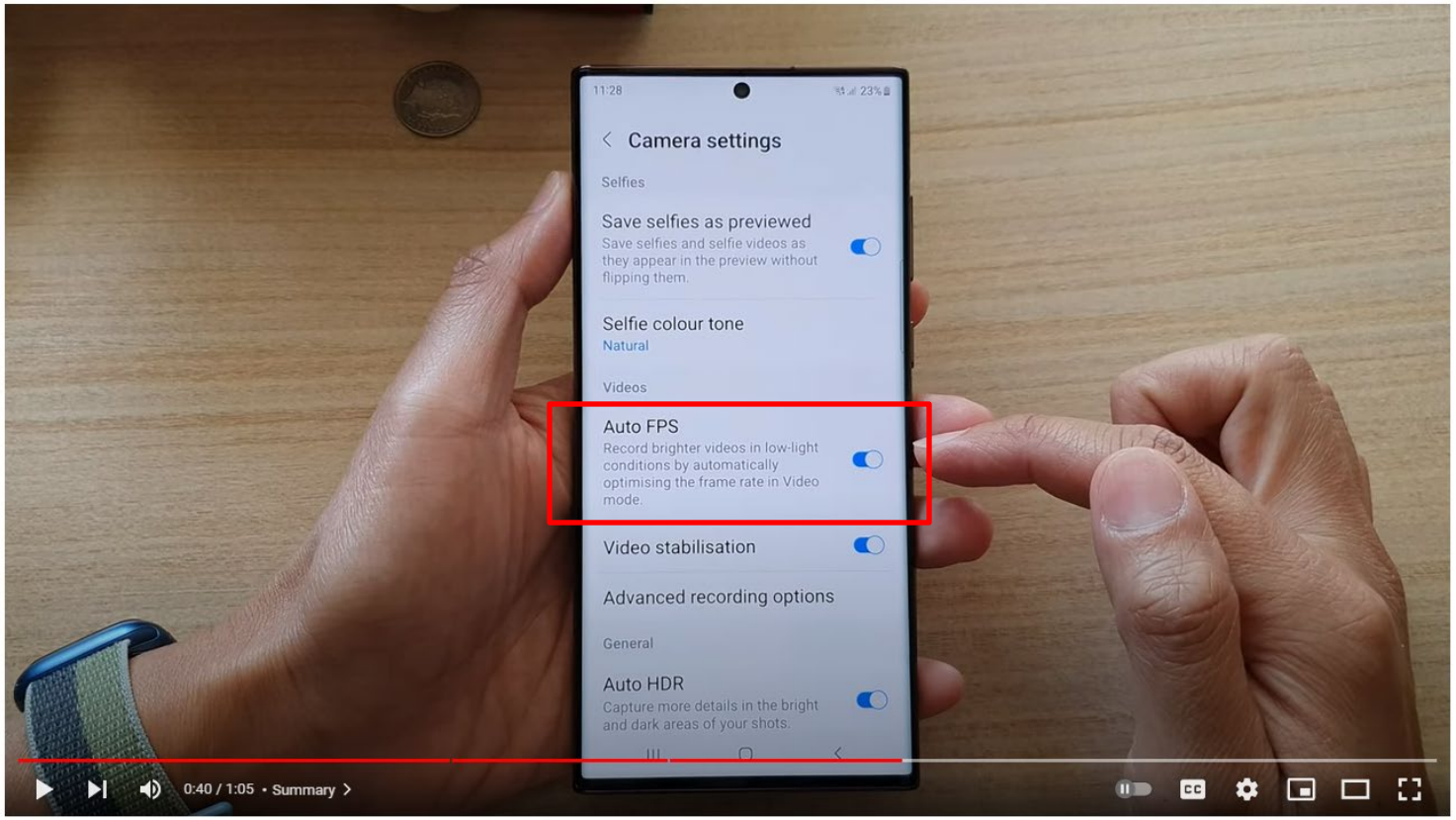
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 – Source: Testing  Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 2 – Source: Testing  Video 2 Details Further, the Samsung Galaxy S22 Ultra smartphone's camera is configured to change frame rates under certain settings (e.g., Super slow-mo, Slow motion, Hyperlapse, Auto FPS)

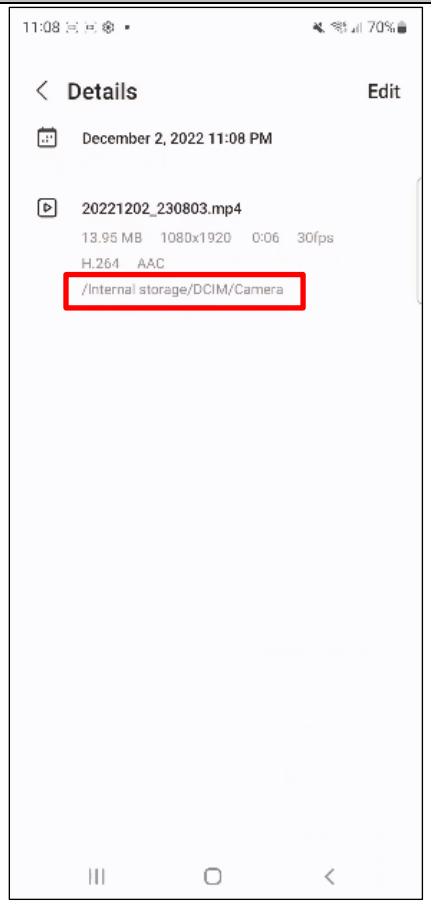
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	- **Super slow-mo**: Record videos at an extremely high frame rate for viewing in high quality slow motion. You can play a specific section of each video in slow motion after recording it. - **Slow motion**: Record videos at a high frame rate for viewing in slow motion. - **Hyperlapse**: Create a time lapse video by recording at various frame rates. The frame rate is adjusted depending on the scene being recorded and the movement of the device. https://downloadcenter.samsung.com/content/UM/202303/20230320235240461/SAM_S901_S906_S908_EN_UM_OS13_012323_FINAL_AC.pdf at 59 **Videos** • **Auto FPS**: Record brighter videos in low-light conditions by automatically optimizing the frame rate in Video mode. https://downloadcenter.samsung.com/content/UM/202303/20230320235240461/SAM_S901_S906_S908_EN_UM_OS13_012323_FINAL_AC.pdf at 61 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1B] an input frame rate	The Samsung Accused Products comprise: An input frame rate specification unit configured to specify the first frame rate of the first moving image data and the second frame rate of the second moving image data.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
specification unit configured to specify the first frame rate of the first moving image data and the second frame rate of the second moving image data,	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone's video camera comprises a frame rate specification option which is an input frame rate specification unit configured to specify (*e.g.*, through user input in the video camera) the first frame rate of the first moving image data and the second frame rate of the second moving image data (*e.g.*, 30/60/Auto frames per second).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 How to Change Video Resultion on SAMSUNG Galaxy S22 Ultra - Set Up Video Resolution https://www.youtube.com/watch?v=oL1JQ4uzFDg

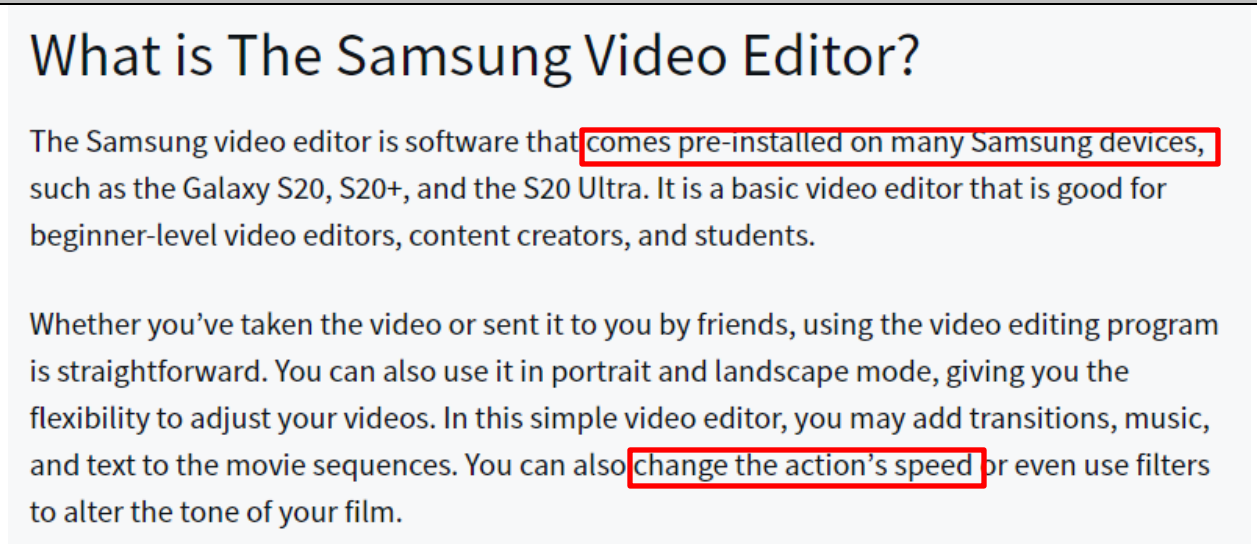
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Galaxy S22/S22+/Ultra: How to Enable/Disable Camera Auto FPS https://www.youtube.com/watch?v=8lEh9gVLagI Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

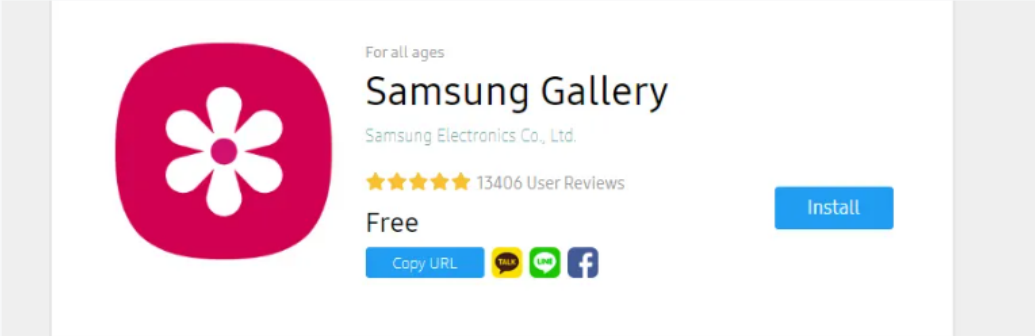
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1C] the first and second moving image data having been input into the moving image data input unit;	In the Samsung Accused Products, the first and second moving image data having been input into the moving image data input unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the first and second moving image data (*e.g.*, Video 1 and Video 2) are input (*e.g.*, captured by the camera) into the moving image data input unit (*e.g.*, Samsung Galaxy S22 Ultra smartphone camera).

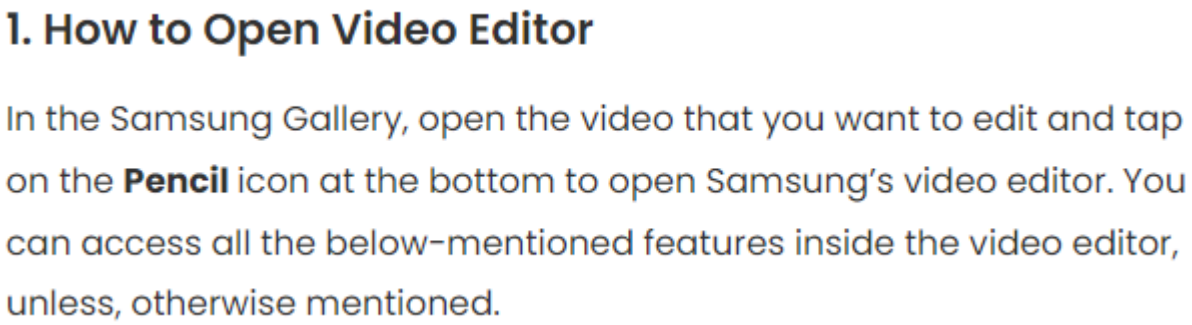
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 – Source: Testing  Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 2 – Source: Testing  Video 2 Details

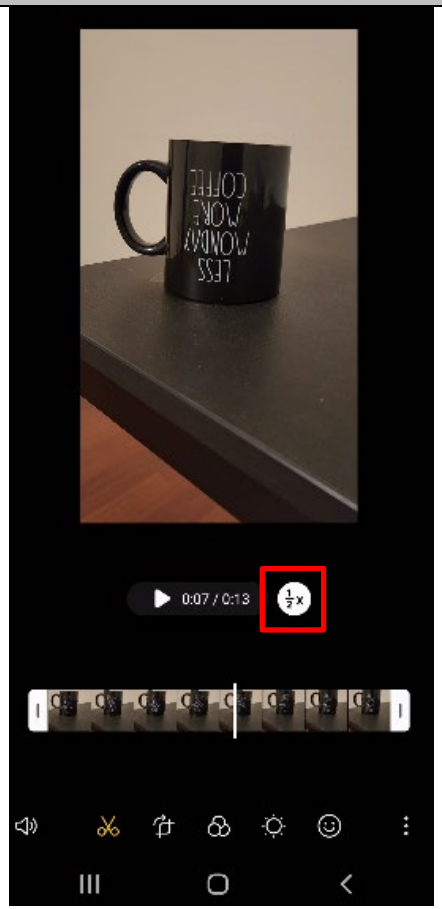
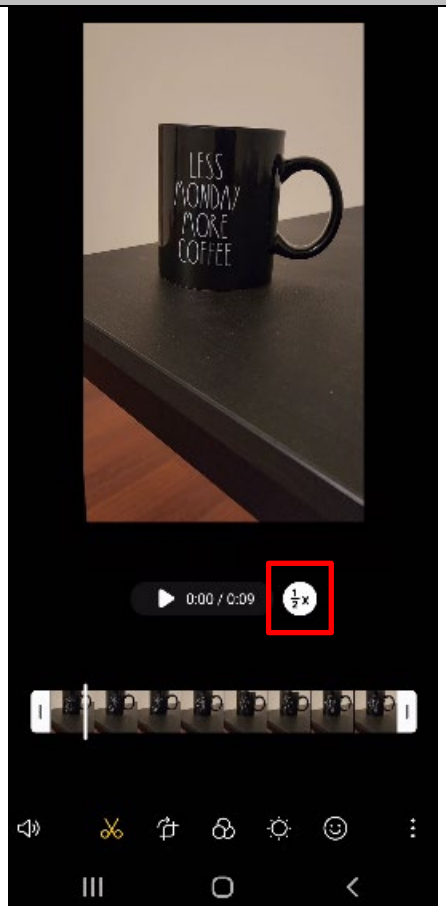
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1D] an output frame rate specification unit configured to specify output frame rate of the first moving image data and output frame rate of the second moving image data that have been input into the moving image data input unit;	The Samsung Accused Products comprise: An output frame rate specification unit configured to specify output frame rate of the first moving image data and output frame rate of the second moving image data that have been input into the moving image data input unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone is an output frame rate specification unit, that comes pre-installed on Samsung devices and is accessible through Samsung Gallery which is also pre-installed on Samsung devices. It is configured to specify (*i.e.*, through user input) output frame rate (*e.g.*, frame rate in which the edited video will be saved) of the first moving image data (*e.g.*, Video 1) and output frame rate of the second moving image data (*e.g.*, Video 2) that have been input into the moving image data input unit (*e.g.*, Samsung Galaxy S22 Ultra smartphone camera).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	# How Video Mixture Software Works: How to Use The Samsung Video Editor? Now that we've gone over some of the best features of the Samsung video editor, let's discuss how to use this video mixture software. Using the Samsung video editor is simple and easy for beginner-level video editors or content creators. Using an editing software like this should only take a To start, open the "Gallery" app and select the video you want to edit. Once the video is open, you will see your available videos to edit.  • Select the video you want to edit, select the "Pencil," and the Samsung Video Editor will come up. https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

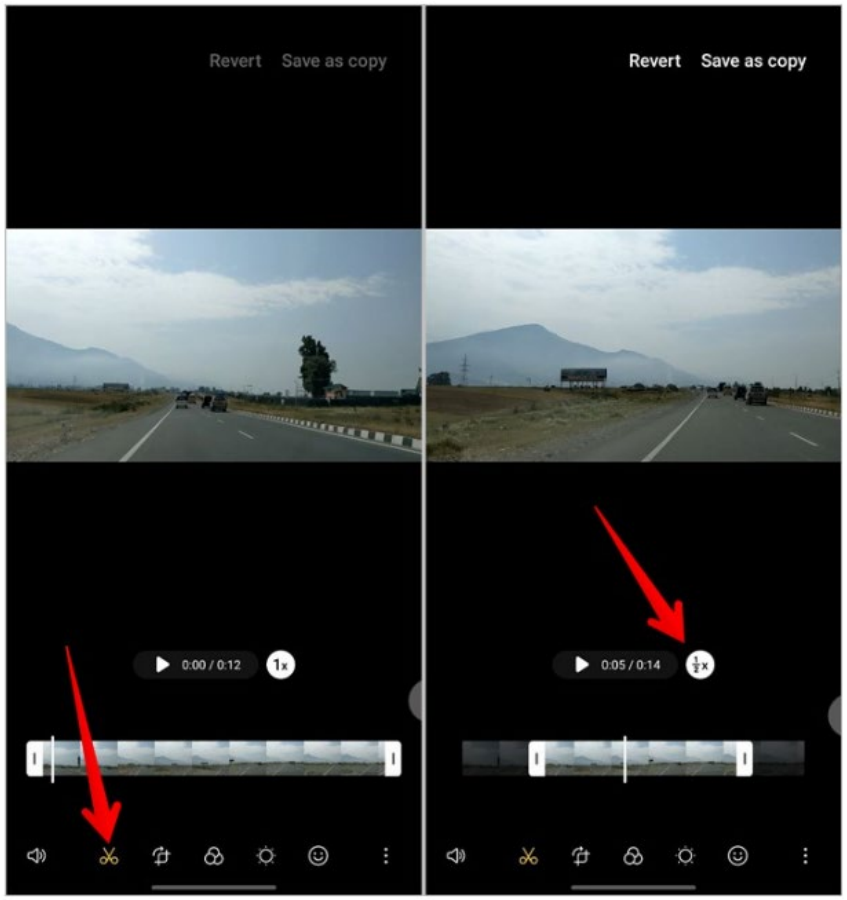
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://techwiser.com/edit-videos-samsung-gallery/ The Samsung Video Editor enables a user to specify playback time, which automatically adjusts the output frame rate of any input video using an output frame rate specification unit. One example shown below is using the Super Slow-mo function where a user can input (*i.e.*, record) a video captured at 480 fps and then output (*i.e.*, digitally enhance) the video to 960 fps. *Super Slow-mo only supports HD resolution. On Galaxy S22, S22+ and S22 Ultra, users can record approximately 0.5 seconds of video captured at 960 fps with approximately 16 seconds of playback. On Galaxy S22 Series, users can record approximately 0.5 second of video captured at 480 fps and digitally enhance the video to 960 fps with approximately 16 seconds of playback. Playback time can be edited in Super Slow-mo player. https://www.samsung.com/global/galaxy/galaxy-s22-ultra/specs/ Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1E] a playback speed specification	The Samsung Accused Products comprise: A playback speed specification unit configured to specify a same playback speed for the first moving image data and the second moving image data.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
unit configured to specify a same playback speed for the first moving image data and the second moving image data;	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor on the Samsung Galaxy S22 Ultra smartphone is a playback speed specification unit configured to specify (*i.e.*, based on user input) a same playback speed (*e.g.*, ½x speed, 1x speed, or 2x speed) for the first moving image data (*e.g.*, Video 1) and the second moving image data (*e.g.*, Video 2).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 in Samsung Video Editor @ ½ Speed Source: Testing  Video 2 in Samsung Video Editor @ ½ Speed Source: Testing

3. How to Change Speed of Video

You can speed up or slow down a video using the Samsung video editor. Press the **Scissor** tool at the bottom. Then, tap on the **1x** icon to cycle through the available speed options i.e., 2x and 1/2x.



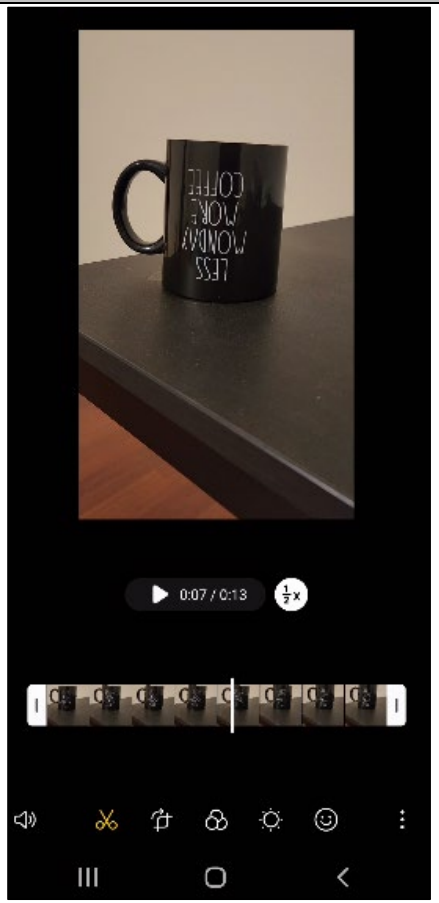
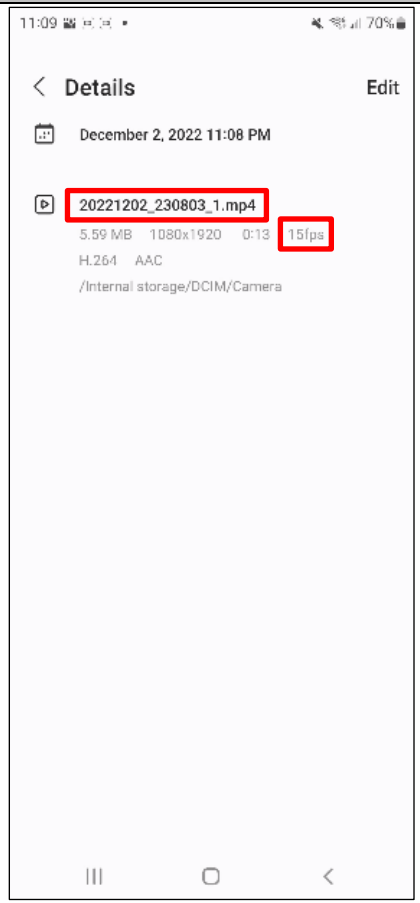
https://techwiser.com/edit-videos-samsung-gallery/

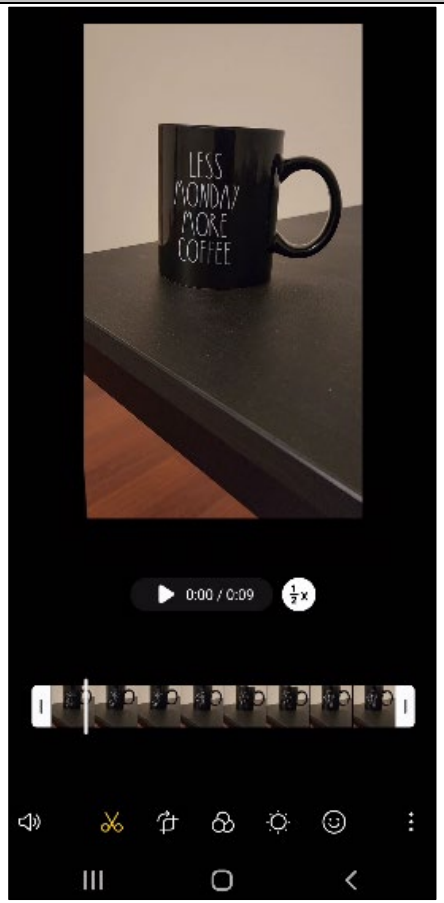
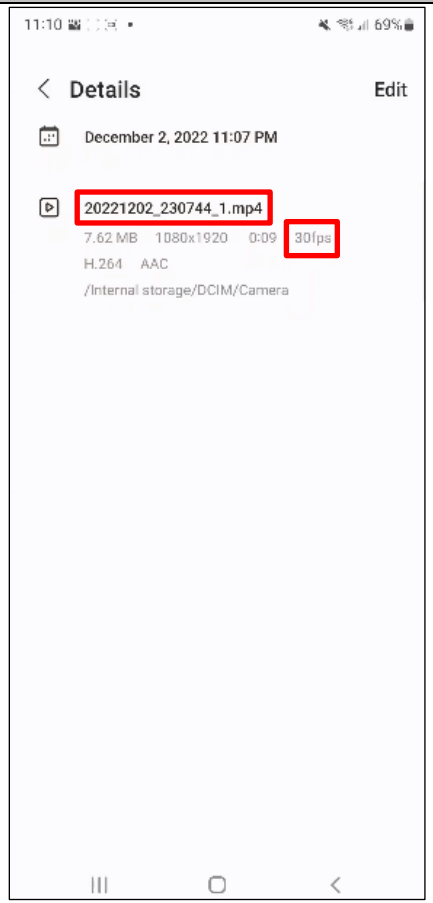
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1F] a frame rate conversion unit configured to change a frame rate of the first moving image data based on the first frame rate, the output frame rate, and the playback speed, and change a frame rate of the second moving image data based on the second frame rate, the output frame rate, and the playback speed; and	The Samsung Accused Products comprise: a frame rate conversion unit configured to change a frame rate of the first moving image data based on the first frame rate, the output frame rate, and the playback speed, and change a frame rate of the second moving image data based on the second frame rate, the output frame rate, and the playback speed. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone is a frame rate conversion unit configured to change the input frame rate to a specified output frame rate based on the user-specified playback speed. As an example, Test Case "i" below shows that the frame rate conversion unit changes the first input frame rate of 30 fps with a user-specified playback speed of .5x to a converted output frame rate of 15fps. Test Case "j" below shows that the frame rate conversion unit changes the second input frame rate of 60 fps with a user-specified playback speed of .5x to a converted output frame rate of 30 fps

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products

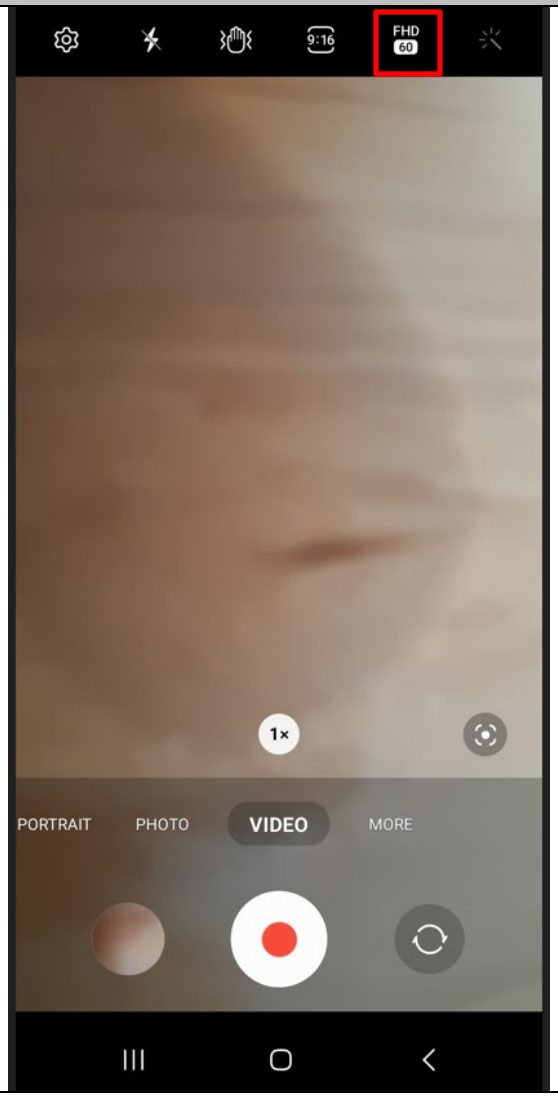
Test Case	Example	Input Frame Rate	Playback Speed	Output Frame Rate
a.		30 fps	1x	30 fps
b.		30 fps	2x	30 fps
c.		60 fps	1x	60 fps
d.		60 fps	2x	60 fps
e.		119 fps	1x	119 fps
f.		119 fps	0.5x	60 fps
g.		119 fps	2x	120 fps
h.		21 fps	2x	30 fps
i.	Video 1	30 fps	0.5x	15 fps
j.	Video 2	60 fps	0.5x	30 fps

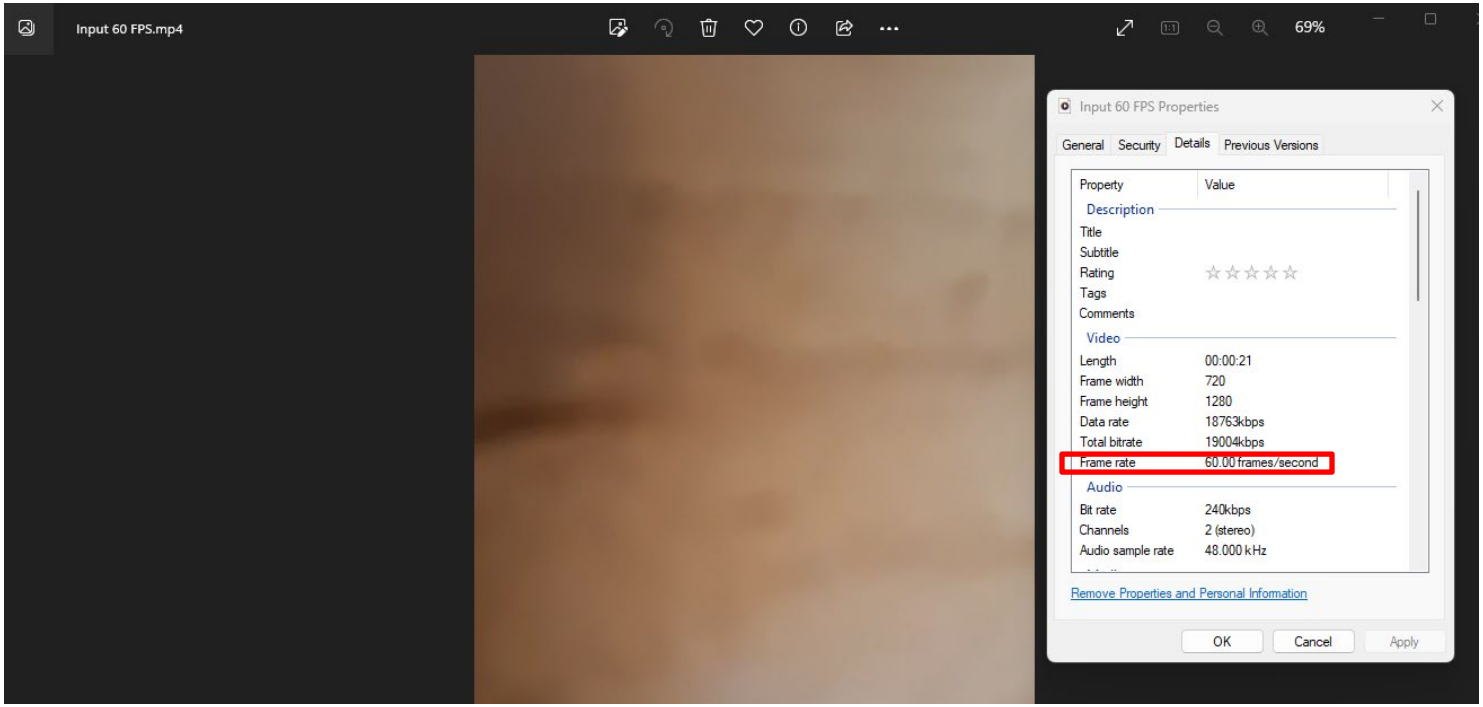
Source: Testing

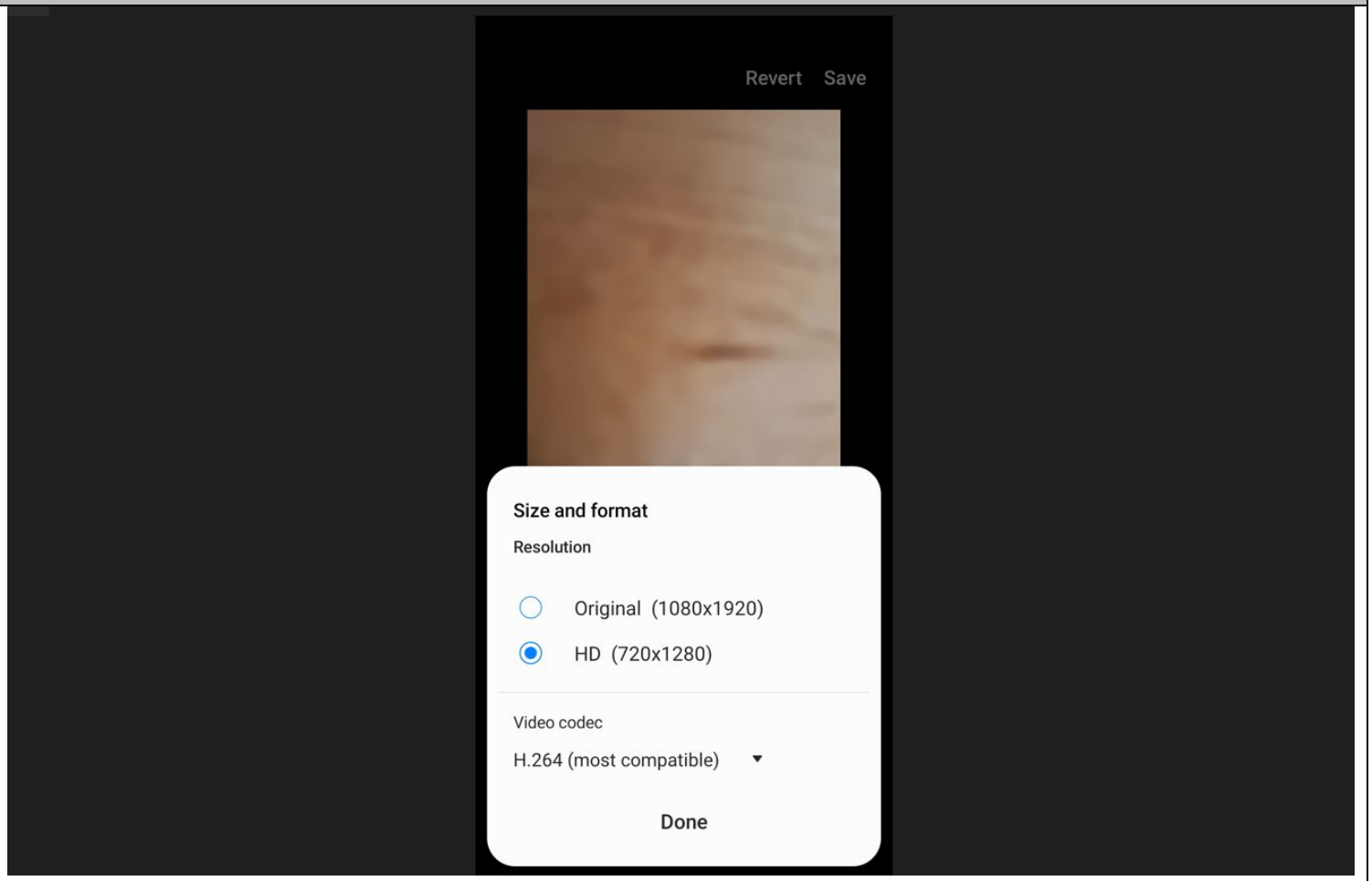
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Converted Video 1 – Source: Testing  Converted Video 1 Details

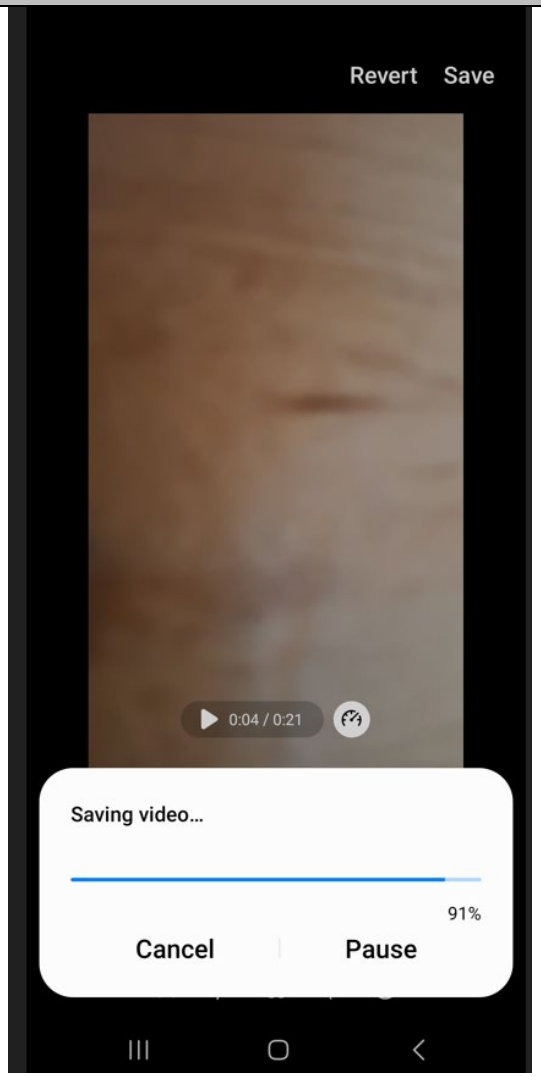
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Converted Video 2 – Source: Testing  Converted Video 2 Details For example, using Super Slow-mo, a frame rate conversion unit (*i.e.*, "frame rate conversion tech") is configured to change the frame rate of a recorded .5 second video captured at 480 fps into a 16 second video captured at 960 fps. That is 1/32 the playback speed but double the frame rate.

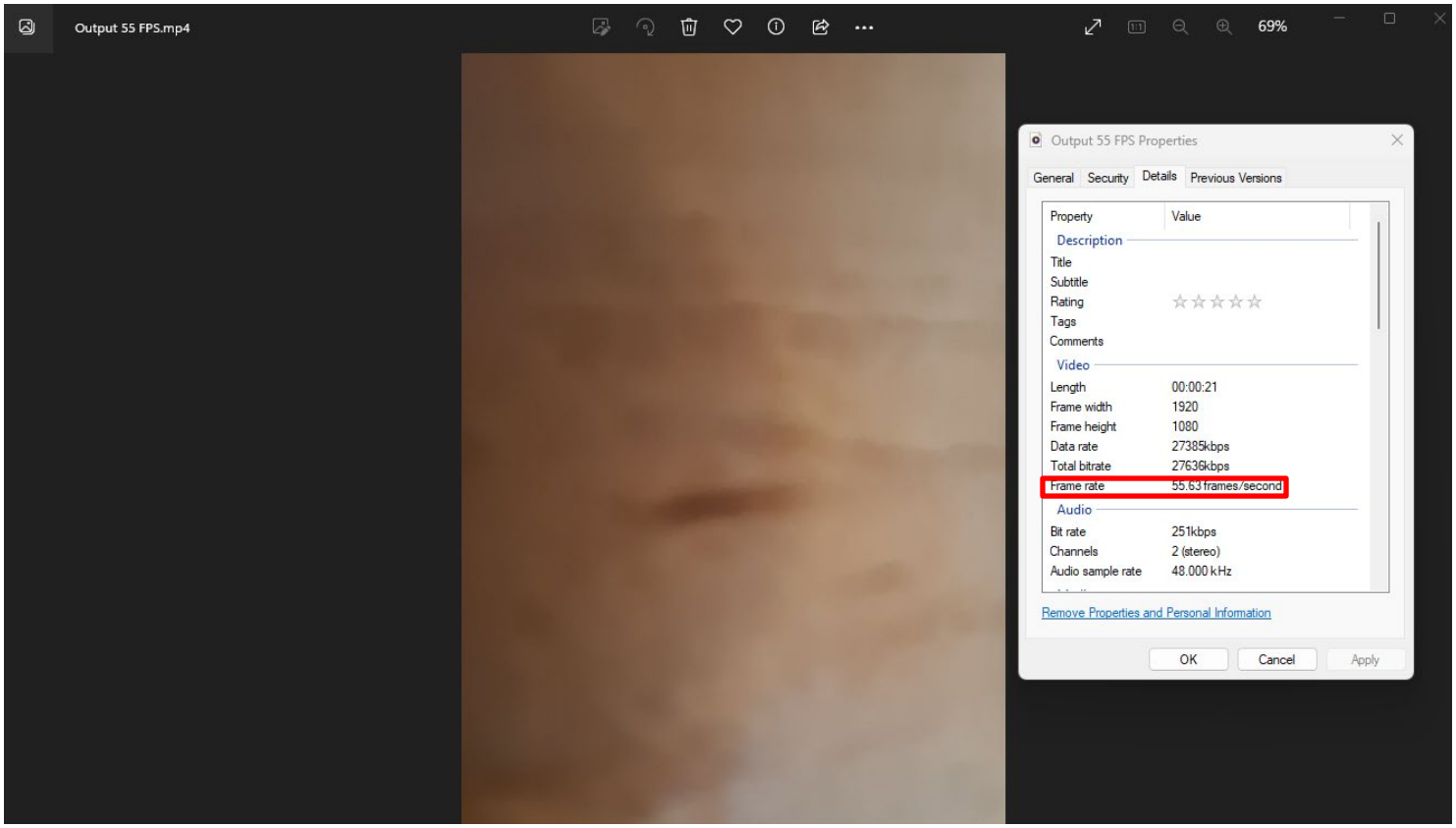
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	*Super Slow-mo only supports HD resolution. On Galaxy S22, S22+ and S22 Ultra, users can record approximately 0.5 seconds of video captured at 960 fps with approximately 16 seconds of playback. On Galaxy S22 Series, users can record approximately 0.5 second of video captured at 480 fps and digitally enhance the video to 960 fps with approximately 16 seconds of playback. Playback time can be edited in Super Slow-mo player. https://www.samsung.com/global/galaxy/galaxy-s22-ultra/specs/ As a little refresher, Slow Motion lets you play a recorded video up to 8 times slower than normal and its amped-up version, Super Slow-mo, plays clips 32 times slower than normal videos. This makes videos appear more artistic and could also be used for adding emphasis. The regular and Plus S22 variants can record 960 fps natively, whereas the S22 Ultra can only record 480fps super slow-motion videos, and then uses the frame rate conversion tech to reach 960fps. https://www.phonearena.com/news/samsung-galaxy-s22-ultra-480fps-slow-motion-video-not-supported_id139224 A frame rate can also be converted by adjusting the output frame rate through resolution settings. In the example below, the input frame rate chosen by selecting FHD, is 60 fps.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	

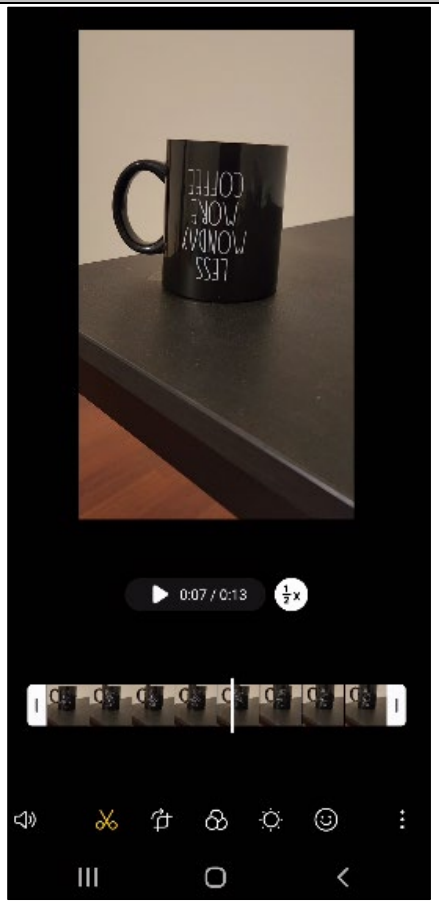
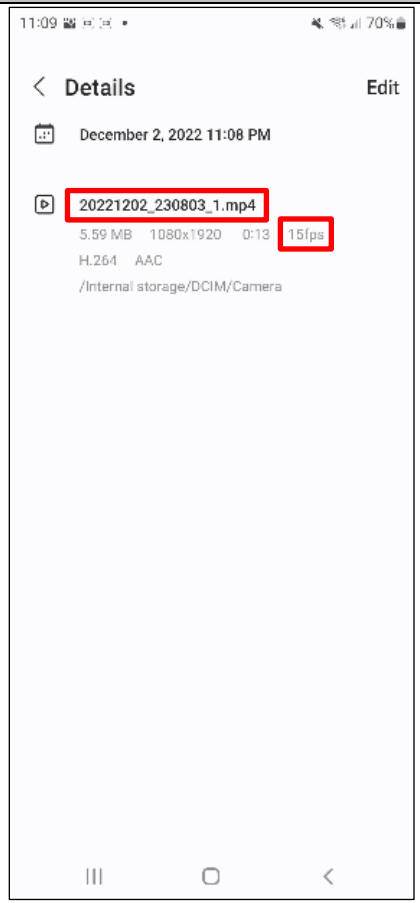
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Source: Testing  Source: Testing By changing the resolution from "Original, (1080x1920)" to "HD (729x1280)," and then saving the resulting video, the output frame rate is converted from the input frame rate of 60 fps to 55.63 fps.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing

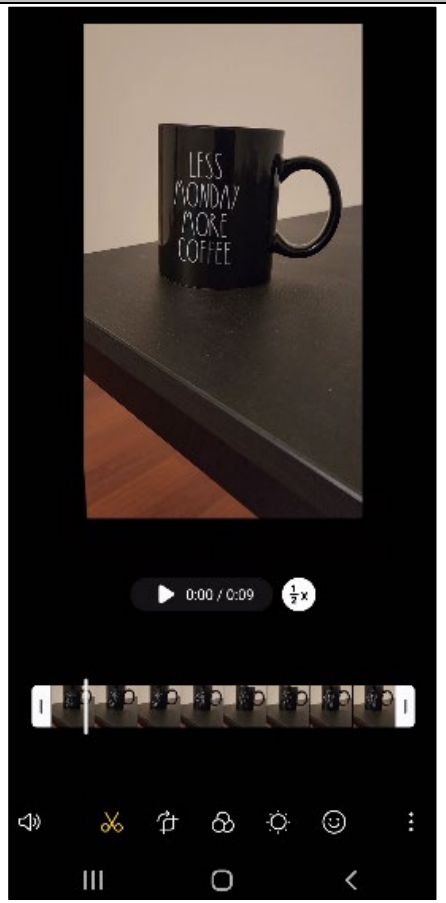
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

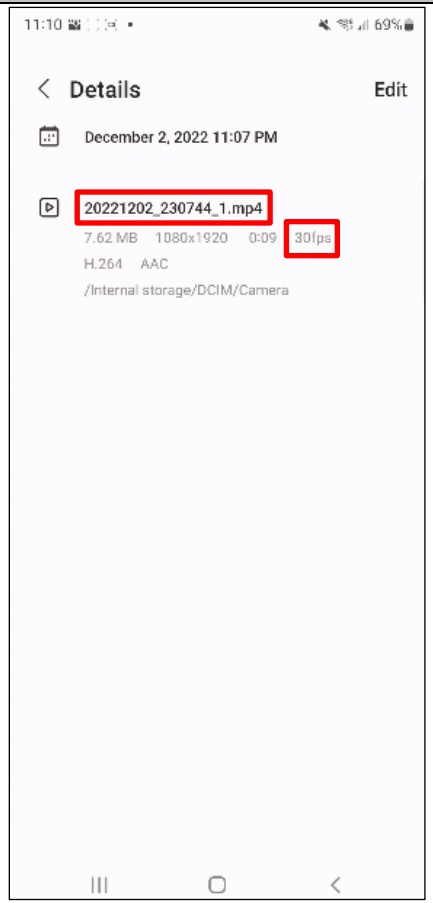
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1G] a moving image data output unit that outputs the first and second moving image data whose frame rates have been converted by the frame rate conversion unit.	The Samsung Accused Products comprise: A moving image data output unit that outputs the first and second moving image data whose frame rates have been converted by the frame rate conversion unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Gallery in combination with the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone is a moving image data output unit that outputs (*e.g.*, creates a new .mp4 video file) the first and second moving image data whose frame rates have been converted (*e.g.*, the frame rate of Video 1 was converted from 30 frames per second to 15 frames per second, the frame rate of Video 2 was converted from 60 frames per second to 30 frames per second) by the frame rate conversion unit (*e.g.*, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone).Both have a playback speed of 0.5x.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Converted Video 1 – Source: Testing  Converted Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products



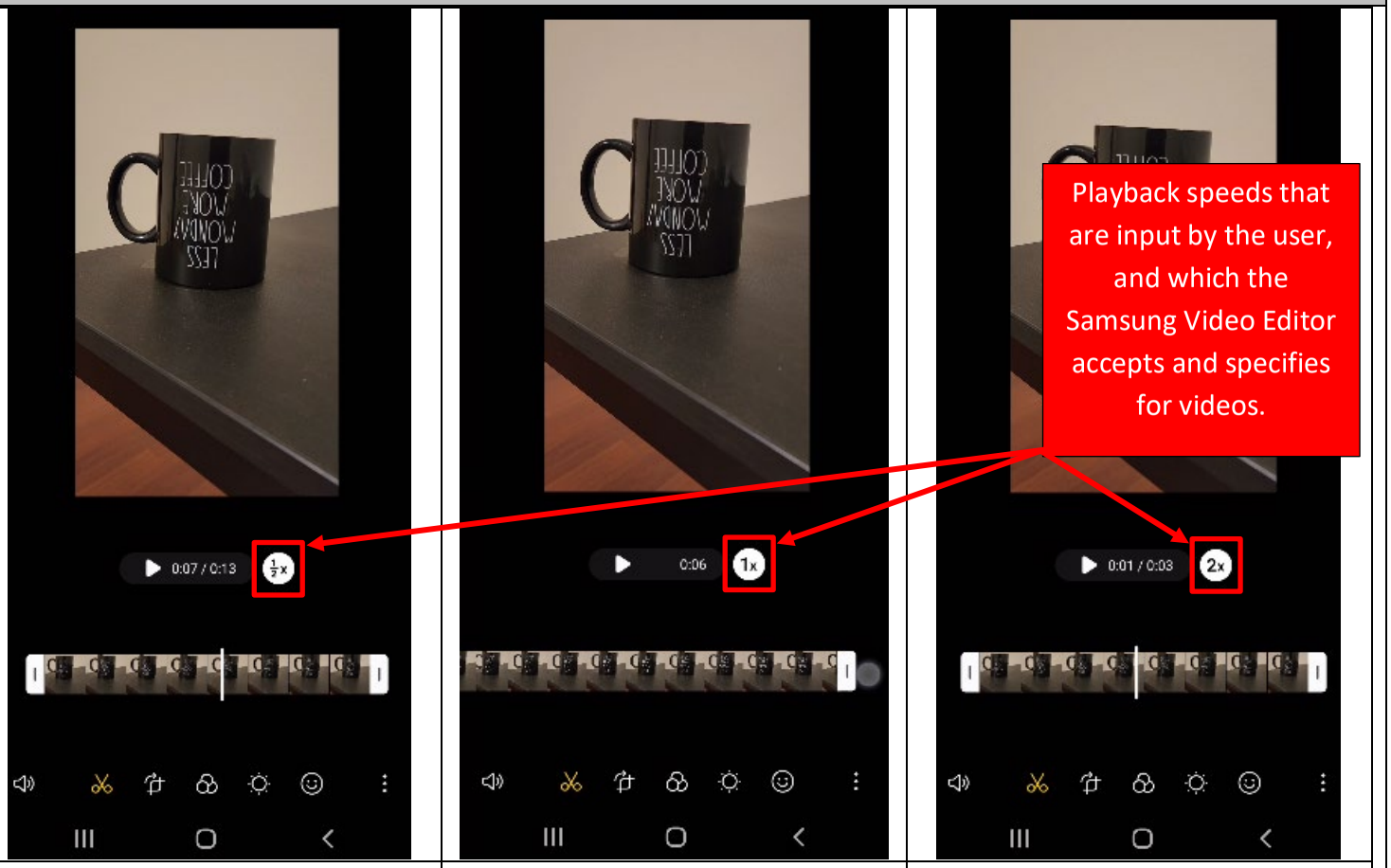
Converted Video 2 – Source: Testing



Converted Video 2 Details

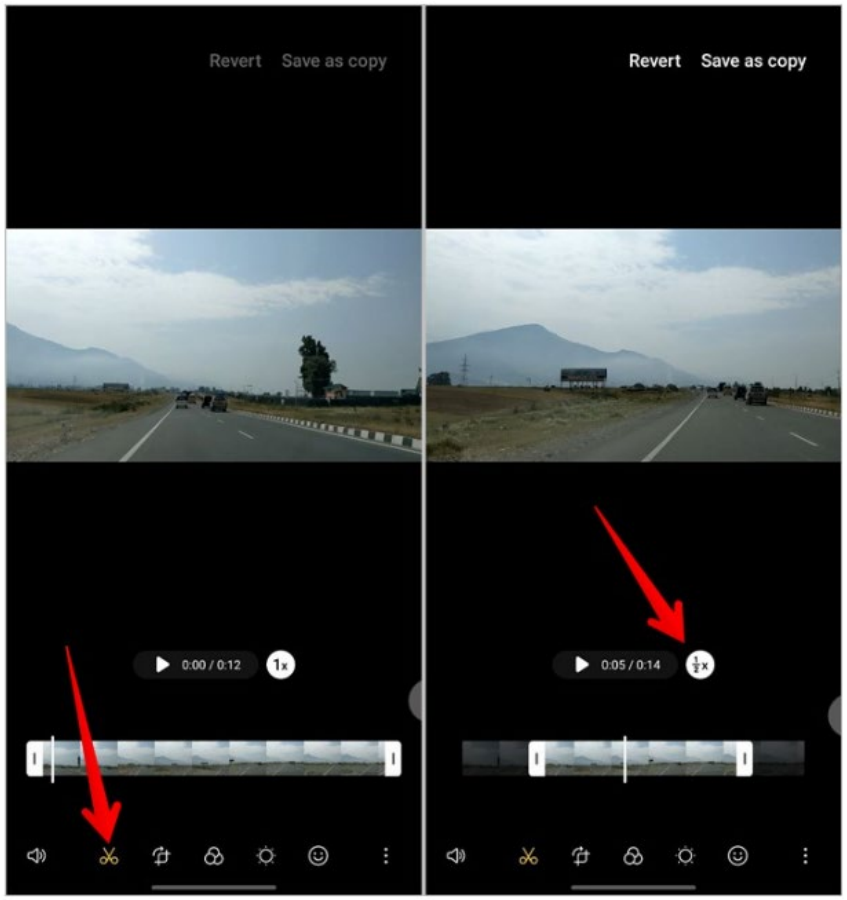
Example	Input Frame Rate	Playback Speed	Changed Frame Rate

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	<table><tr><td>Video 1</td><td>30 fps</td><td>0.5x</td><td>15 fps</td></tr><tr><td>Video 2</td><td>60 fps</td><td>0.5x</td><td>30 fps</td></tr></table> Source: Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2] The moving image data processing apparatus according to claim 1, further comprising an operating unit configured to accept an input to specify the playback speed, wherein the playback speed specification unit specifies the playback speed	The Samsung Accused Products comprise: An operating unit configured to accept an input to specify the playback speed, wherein the playback speed specification unit specifies the playback speed based on the input that is accepted by the operating unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone comprises an operating unit configured to accept an input (*e.g.*, from a user selecting a playback speed using the selectable playback speed button) to specify the playback speed (*e.g.*, selectable playback speed button indicating ½x, 1x and 2x playback speeds), wherein the playback speed specification unit specifies the playback speed based on the input that is accepted by the operating unit.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
based on the input that is accepted by the operating unit.	
	Video 1 in Samsung Video Editor @ ½x Speed Source: Testing Video 1 in Samsung Video Editor @ 1x Speed Source: Testing Video 1 in Samsung Video Editor @ 2x Speed Source: Testing

3. How to Change Speed of Video

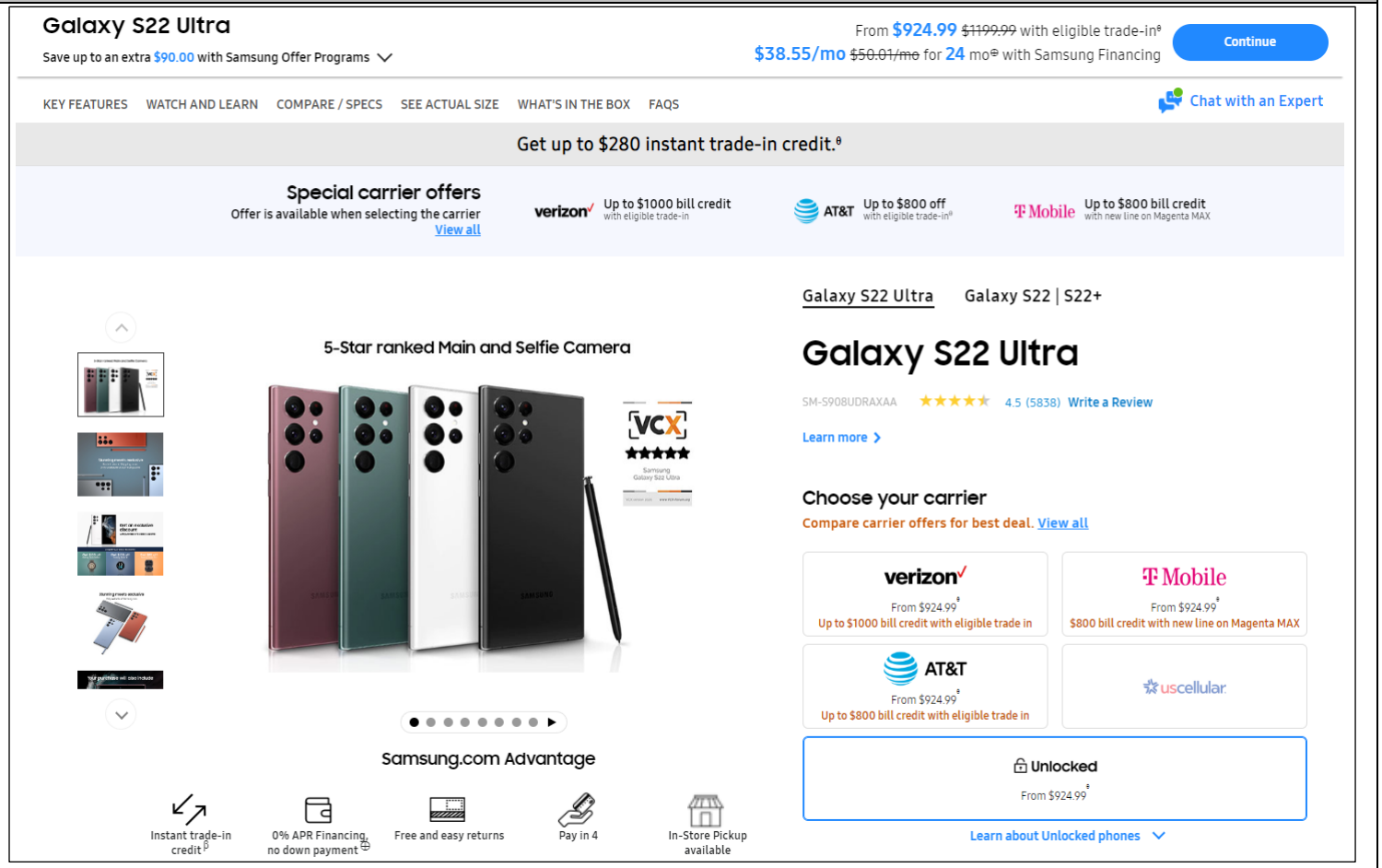
You can speed up or slow down a video using the Samsung video editor. Press the **Scissor** tool at the bottom. Then, tap on the **1x** icon to cycle through the available speed options i.e., 2x and 1/2x.



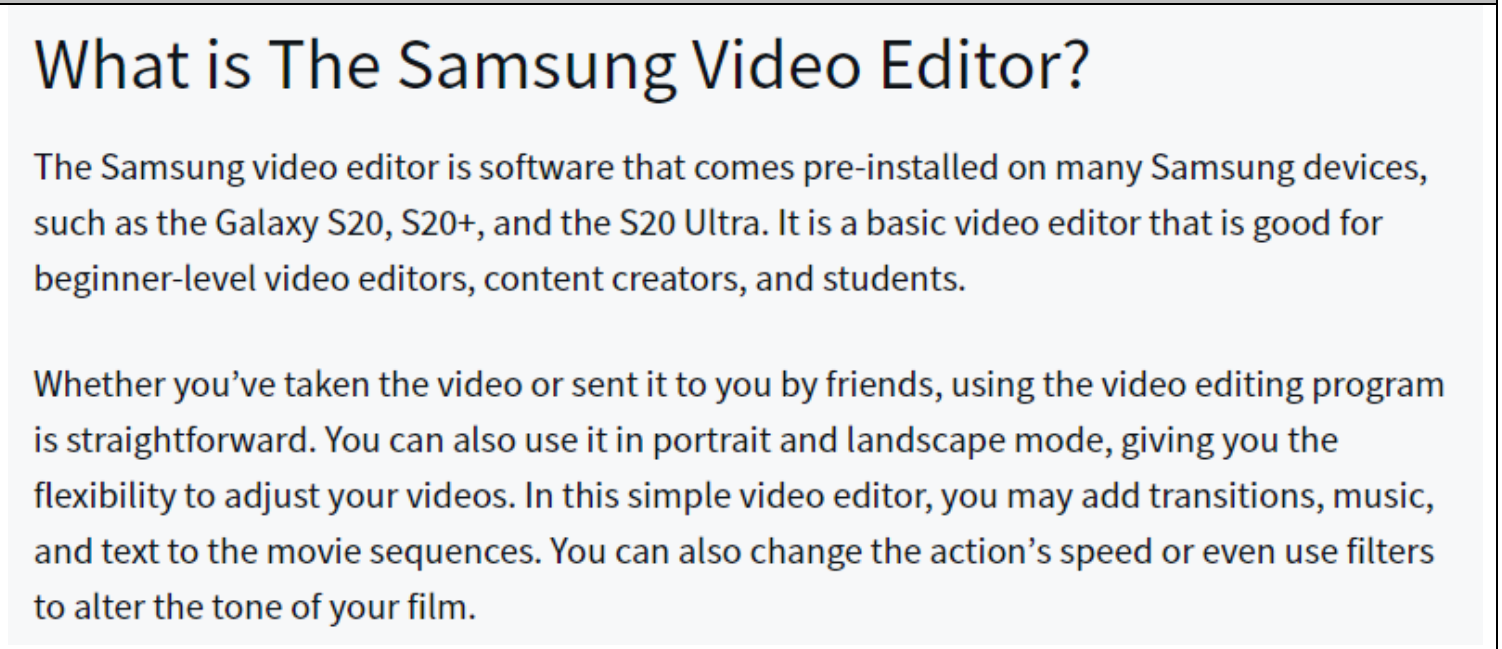
https://techwiser.com/edit-videos-samsung-gallery/

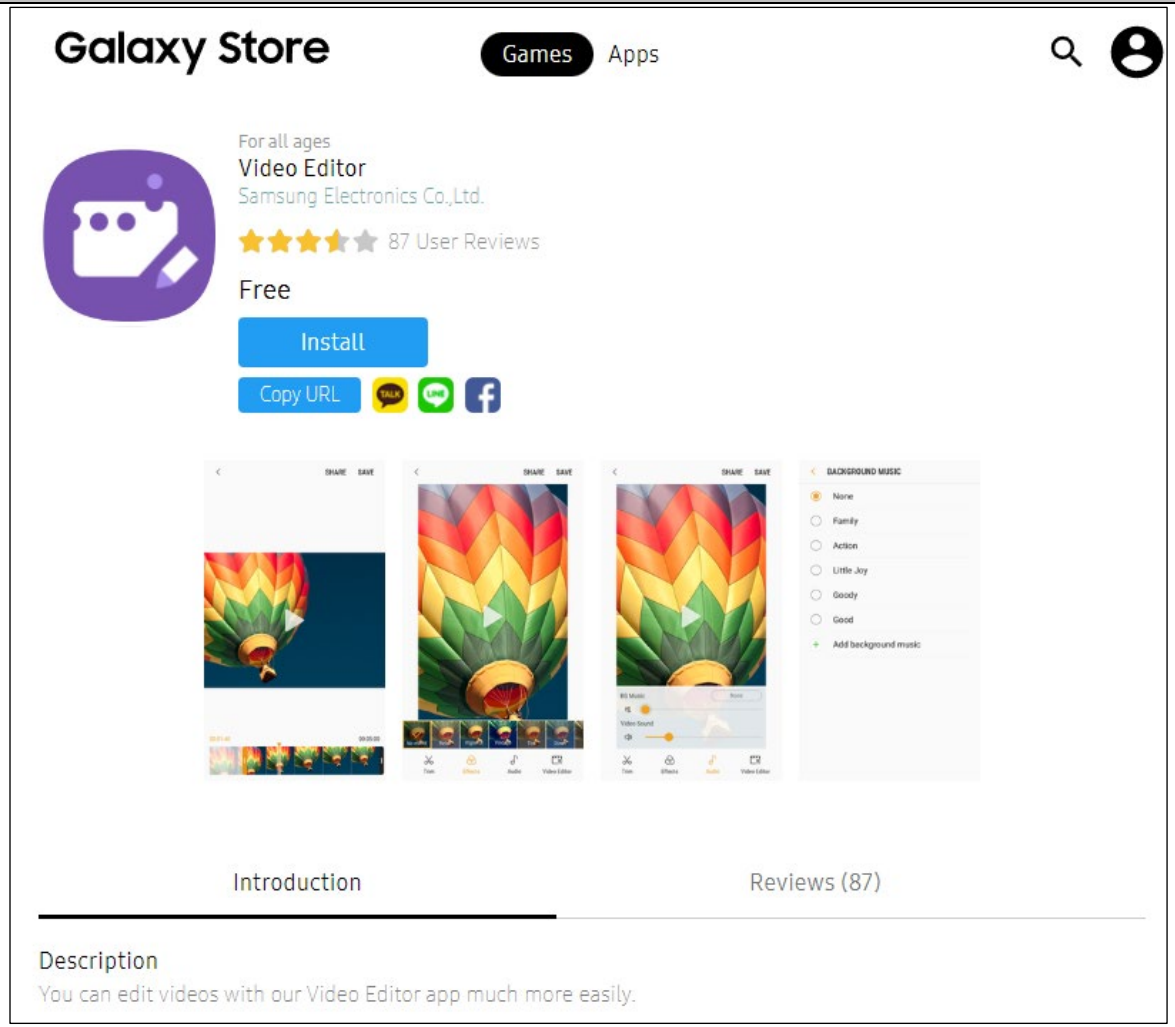
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4] The moving image data processing apparatus according to claim 1, wherein the output frame rate specification unit is configured to specify the output frame rate based on information of an external device receiving the moving image data output by the moving image data output unit.	In the Samsung Accused Products, the output frame rate specification unit is configured to specify the output frame rate based on information of an external device receiving the moving image data output by the moving image data output unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, in the Samsung Video Editor in the Samsung Galaxy S22 Ultra, the output frame rate specification unit is configured to specify the output frame rate (*e.g.*, controlled playback through accessing a Media panel) based on information of an external device receiving the moving image data output (*e.g.*, connected video devices or devices connected to supported apps like SmartThings or Google Home) by the moving image data output unit (*e.g.*, the Samsung Video Editor output unit). • **Device control**: Control other devices when supported apps like SmartThings or Google Home are installed. • **Media output**: Access the Media panel and control playback of connected audio and video devices. https://downloadcenter.samsung.com/content/UM/202303/20230320235240461/SAM_S901_S906_S908_EN_UM_OS13_012323_FINAL_AC.pdf at 29

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[6P]. A moving image data processing method comprising:	Samsung or users of the Samsung Accused Products perform: A moving image data processing method. Samsung infringes directly, and/or indirectly by providing devices such as smartphones and tablets with the Samsung Video Editor and/or Samsung Gallery installed that are each, or in combination, a moving image data processing apparatus, and encouraging others, including customers and end users, to download the Samsung Video Editor app onto their hardware devices. Samsung also infringes directly or indirectly by providing specific Samsung software updates including the Samsung Video Editor and/or Samsung Gallery app onto hardware devices of their customers and end users. Samsung directly infringes by using hardware devices with the Samsung Video Editor and/or Samsung Gallery app installed, such as during testing and qualification. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone with Samsung Gallery and the Samsung Video Editor processes moving image data, such as videos. On information and belief, Samsung itself tests and uses smartphones and tablets having the Samsung Gallery and Samsung Video Editor app installed. Additionally, Samsung provides the Samsung Gallery and Samsung Video Editor app to customers and users on its smartphones and devices and in its app store.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	

https://www.samsung.com/us/smartphones/galaxy-s22-ultra/buy/galaxy-s22-ultra-128gb-unlocked-sm-s908udraxaa/?modelCode=SM-S908UZAAXAA

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

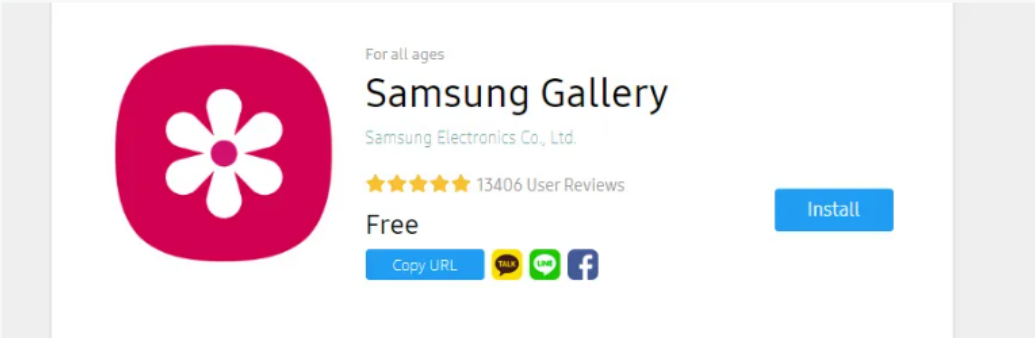
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://galaxystore.samsung.com/prepost/000004444189?langCd=en

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products

How Video Mixture Software Works: How to Use The Samsung Video Editor?

Now that we've gone over some of the best features of the Samsung video editor, let's discuss how to use this video mixture software. Using the Samsung video editor is simple and easy for beginner-level video editors or content creators. Using an editing software like this should only take a

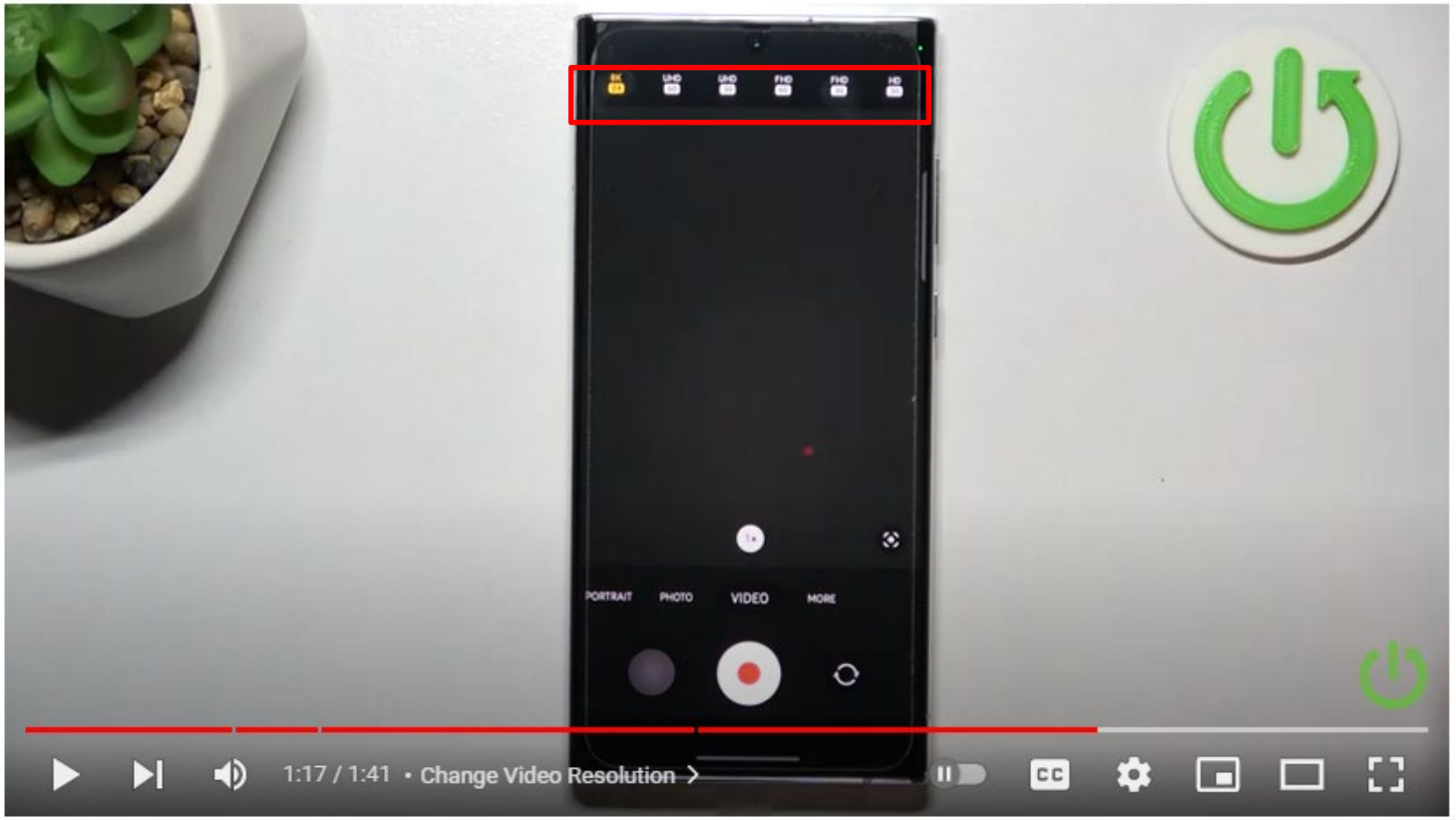
To start, open the "Gallery" app and select the video you want to edit. Once the video is open, you will see your available videos to edit.



- Select the video you want to edit, select the "Pencil," and the Samsung Video Editor will come up.

https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[6A] inputting first moving image data generated at a first frame rate and second image data generated at a second frame rate that is different from the first frame rate;	Samsung or users of the Samsung Accused Products perform: inputting first moving image data generated at a first frame rate and second image data generated at a second frame rate that is different from the first frame rate. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone's video camera is configured to input first moving image data generated at a first frame rate (*e.g.*, Video 1 at 30 frames per second) and second image data generated at a second frame rate that is different from the first frame rate (*e.g.*, Video 2 at 60 frames per second).

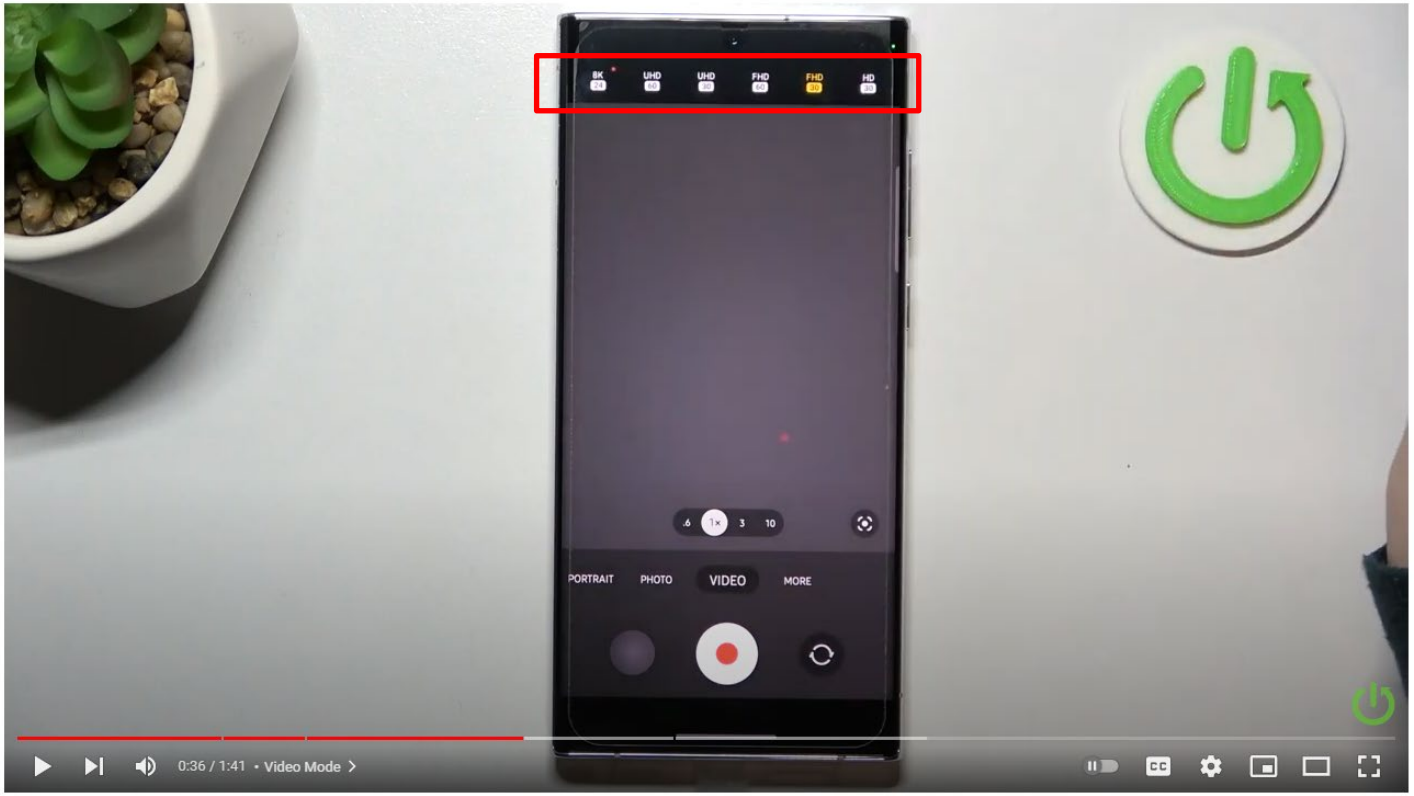
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 How to Change Video Resultion on SAMSUNG Galaxy S22 Ultra - Set Up Video Resolution https://www.youtube.com/watch?v=oL1JQ4uzFDg

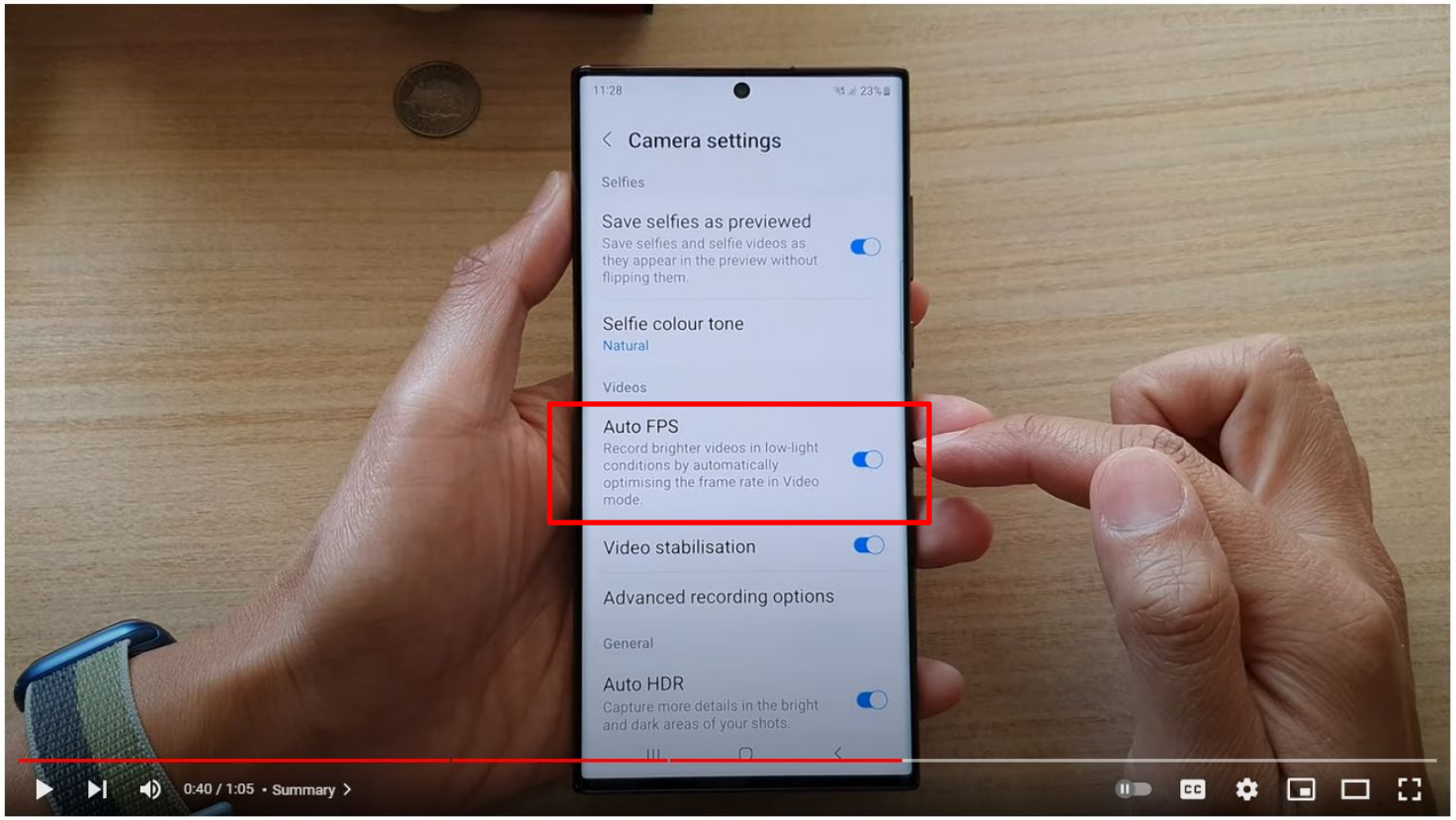
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 – Source: Testing  Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 2 – Source: Testing  Video 2 Details Further, the Samsung Galaxy S22 Ultra smartphone's camera is configured to change frame rates under certain settings (e.g., Super slow-mo, Slow motion, Hyperlapse, Auto FPS)

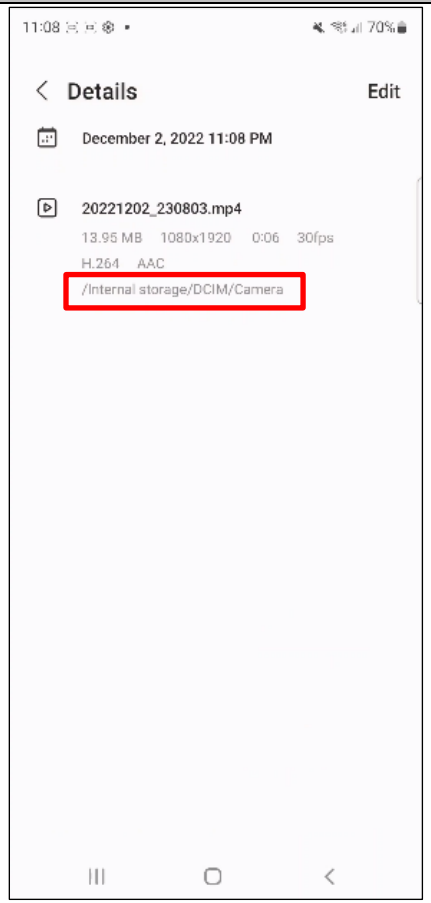
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	- **Super slow-mo**: Record videos at an extremely high frame rate for viewing in high quality slow motion. You can play a specific section of each video in slow motion after recording it. - **Slow motion**: Record videos at a high frame rate for viewing in slow motion. - **Hyperlapse**: Create a time lapse video by recording at various frame rates. The frame rate is adjusted depending on the scene being recorded and the movement of the device. https://downloadcenter.samsung.com/content/UM/202303/20230320235240461/SAM_S901_S906_S908_EN_UM_OS13_012323_FINAL_AC.pdf at 59 **Videos** • **Auto FPS**: Record brighter videos in low-light conditions by automatically optimizing the frame rate in Video mode. https://downloadcenter.samsung.com/content/UM/202303/20230320235240461/SAM_S901_S906_S908_EN_UM_OS13_012323_FINAL_AC.pdf at 61 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[6B] specifying the first frame rate of the first moving image data and the second frame rate of the second moving image data;	Samsung or users of the Samsung Accused Products perform: specifying the first frame rate of the first moving image data and the second frame rate of the second moving image data. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Galaxy S22 Ultra smartphone's video camera specifies (*e.g.*, through user input in the video camera) the first frame rate of the first moving image data and the second frame rate of the second moving image data (*e.g.*, 30/60/Auto frames per second).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 How to Change Video Resultion on SAMSUNG Galaxy S22 Ultra - Set Up Video Resolution https://www.youtube.com/watch?v=oL1JQ4uzFDg

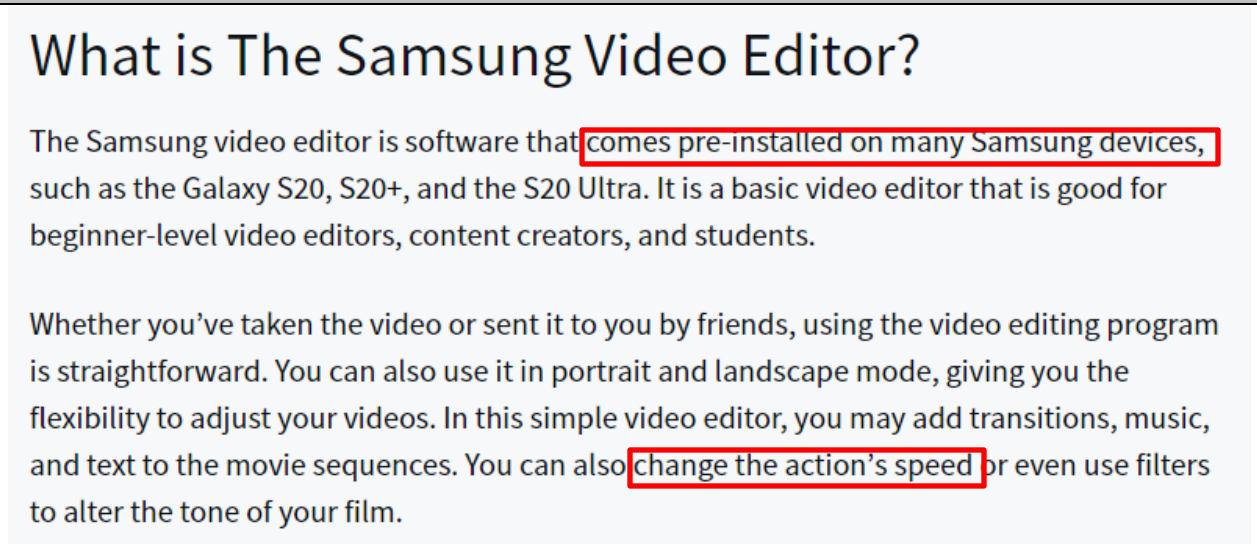
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Galaxy S22/S22+/Ultra: How to Enable/Disable Camera Auto FPS https://www.youtube.com/watch?v=8lEh9gVLagI Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[6C] the first and second moving image data having been input;	In the Samsung Accused Products, the first and second moving image data having been input into the moving image data input unit. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the first and second moving image data (*e.g.*, Video 1 and Video 2) are input (*e.g.*, captured by the camera) into the moving image data input unit (*e.g.*, Samsung Galaxy S22 Ultra smartphone camera).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 – Source: Testing  Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 2 – Source: Testing  Video 2 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[6D] specifying output frame rate of the first moving image data and output frame rate of the second moving image data that have been input	Users of the Samsung Accused Products perform: specifying output frame rate of the first moving image data and output frame rate of the second moving image data that have been input. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone is accessible through Samsung Gallery and specifies (*i.e.*, through user input) output frame rate (*e.g.*, frame rate in which the edited video will be saved) of the first moving image data (*e.g.*, Video 1) and output frame rate of the second moving image data (*e.g.*, Video 2) that have been input into the moving image data input unit (*e.g.*, Samsung Galaxy S22 Ultra smartphone camera).

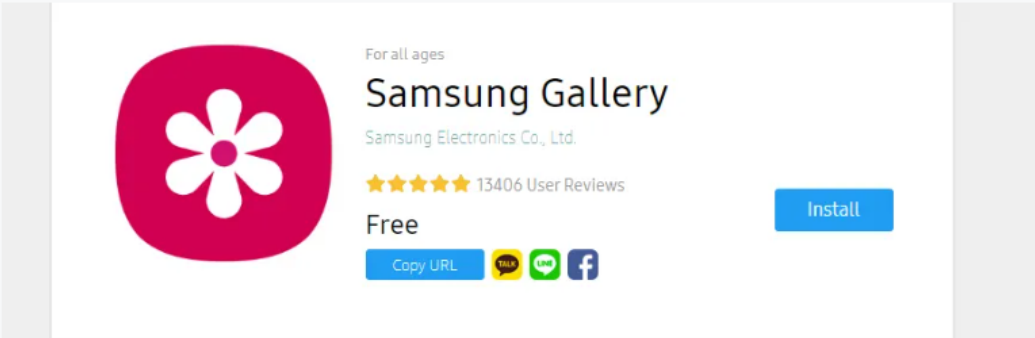
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products

How Video Mixture Software Works: How to Use The Samsung Video Editor?

Now that we've gone over some of the best features of the Samsung video editor, let's discuss how to use this video mixture software. Using the Samsung video editor is simple and easy for beginner-level video editors or content creators. Using an editing software like this should only take a

To start, open the "Gallery" app and select the video you want to edit. Once the video is open, you will see your available videos to edit.

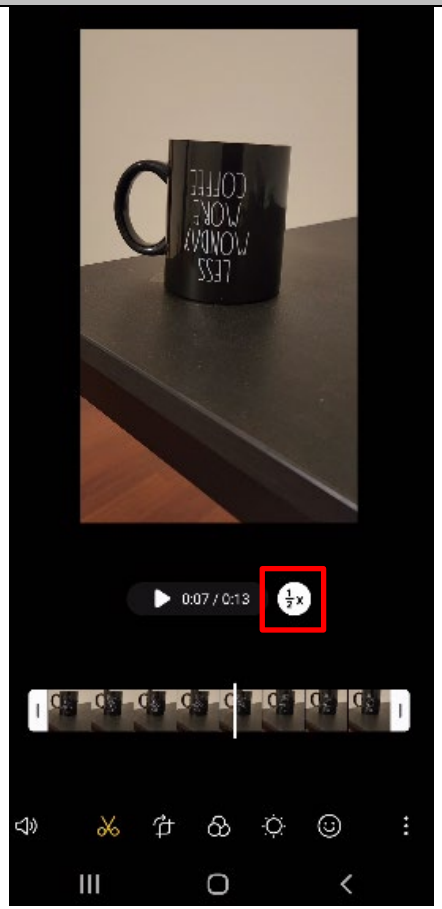
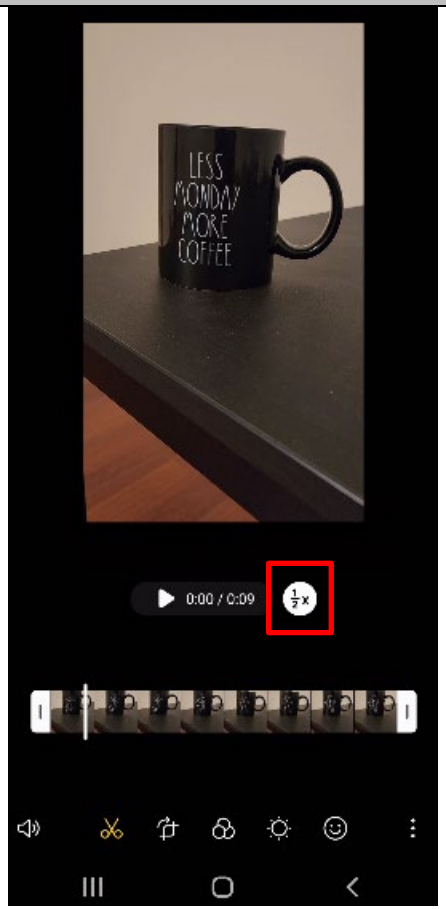


- Select the video you want to edit, select the "Pencil," and the Samsung Video Editor will come up.

https://typecast.ai/learn/how-to-use-the-samsung-video-editor/#:~:text=The%20Samsung%20video%20editor%20is%20software%20that%20comes%20pre%2Dinstalled,%2C%20content%20creators%2C%20and%20students

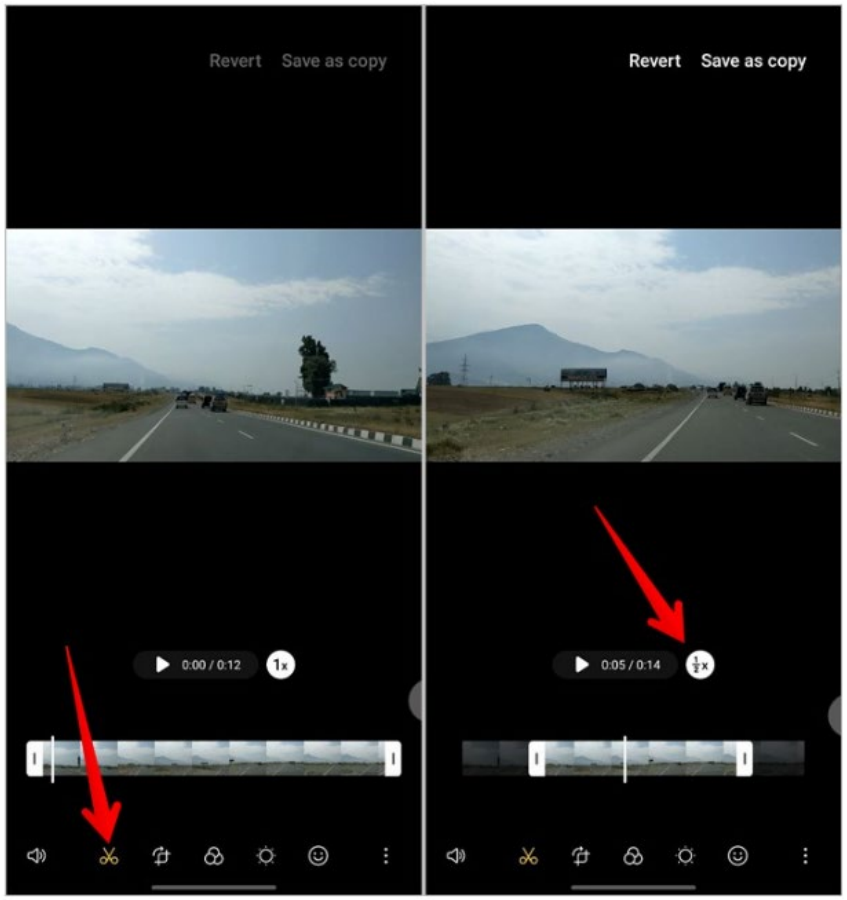
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	1. How to Open Video Editor In the Samsung Gallery, open the video that you want to edit and tap on the **Pencil** icon at the bottom to open Samsung's video editor. You can access all the below-mentioned features inside the video editor, unless, otherwise mentioned. https://techwiser.com/edit-videos-samsung-gallery/ The Samsung Video Editor enables a user to specify playback time, which automatically adjusts the output frame rate of any input video using an output frame rate specification unit. One example shown below is using the Super Slow-mo function where a user can input (i.e., record) a video captured at 480 fps and then output (i.e., digitally enhance) the video to 960 fps. *Super Slow-mo only supports HD resolution. On Galaxy S22, S22+ and S22 Ultra, users can record approximately 0.5 seconds of video captured at 960 fps with approximately 16 seconds of playback. On Galaxy S22 Series, users can record approximately 0.5 second of video captured at 480 fps and digitally enhance the video to 960 fps with approximately 16 seconds of playback. Playback time can be edited in Super Slow-mo player. https://www.samsung.com/global/galaxy/galaxy-s22-ultra/specs/ Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[6E] specifying a same playback speed for the	Users of the Samsung Accused Products perform: specifying a same playback speed for the first moving image data and the second moving image data.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
first moving image data and the second moving image data;	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor on the Samsung Galaxy S22 Ultra smartphone specifies (*i.e.*, based on user input) a same playback speed (*e.g.*, ½x speed, 1x speed, or 2x speed) for the first moving image data (*e.g.*, Video 1) and the second moving image data (*e.g.*, Video 2).

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Video 1 in Samsung Video Editor @ ½ Speed Source: Testing  Video 2 in Samsung Video Editor @ ½ Speed Source: Testing

3. How to Change Speed of Video

You can speed up or slow down a video using the Samsung video editor. Press the **Scissor** tool at the bottom. Then, tap on the **1x** icon to cycle through the available speed options i.e., 2x and 1/2x.



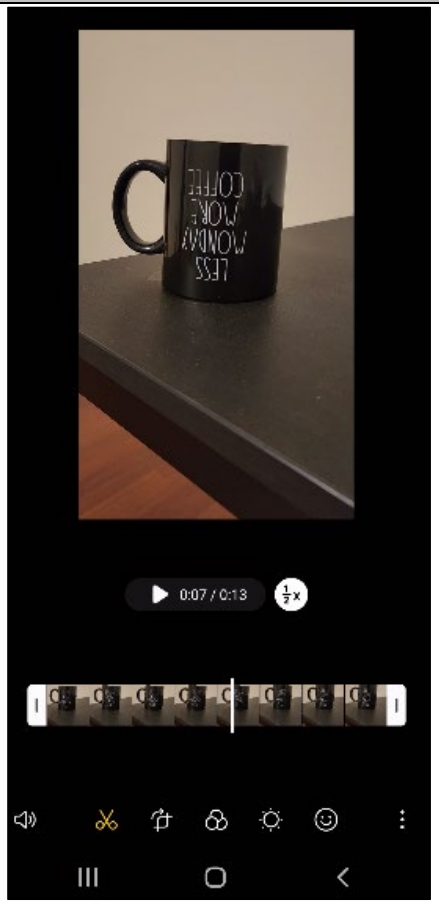
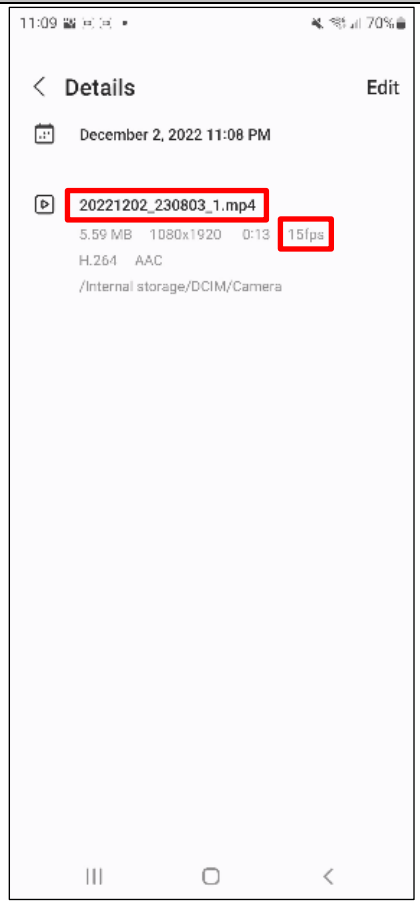
https://techwiser.com/edit-videos-samsung-gallery/

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[6F] changing a frame rate of the first moving image data based on the first frame rate, the output frame rate, and the playback speed, and changing the frame rate of the second moving image data based on the second frame rate, the output frame rate, and the playback speed; and	Users of the Samsung Accused Products perform: changing a frame rate of the first moving image data based on the first frame rate, the output frame rate, and the playback speed, and changing a frame rate of the second moving image data based on the second frame rate, the output frame rate, and the playback speed. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone changes the input frame rate to a specified output frame rate based on the user-specified playback speed. As an example, Test Case "i" below shows that the frame rate conversion unit changes the first input frame rate of 30 fps with a user-specified playback speed of .5x to a converted output frame rate of 15fps. Test Case "j" below shows that the frame rate conversion unit changes the second input frame rate of 60 fps with a user-specified playback speed of .5x to a converted output frame rate of 30 fps

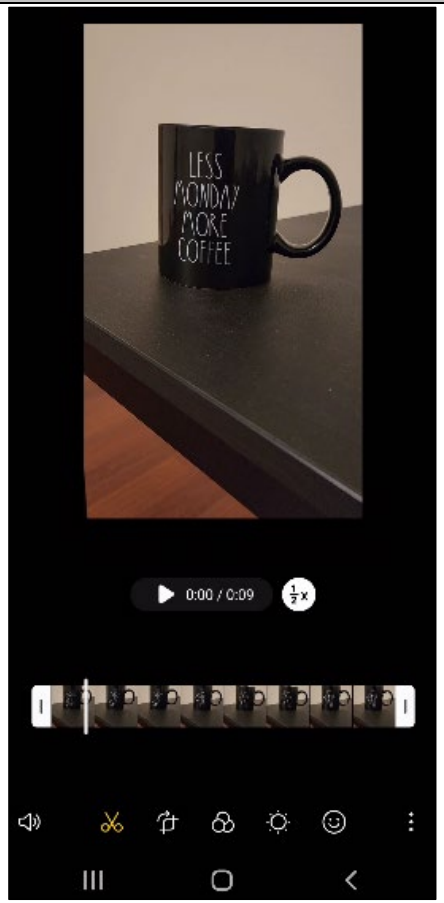
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Test Case	Example	Input Frame Rate	Playback Speed	Output Frame Rate
a.		30 fps	1x	30 fps
b.		30 fps	2x	30 fps
c.		60 fps	1x	60 fps
d.		60 fps	2x	60 fps
e.		119 fps	1x	119 fps
f.		119 fps	0.5x	60 fps
g.		119 fps	2x	120 fps
h.		21 fps	2x	30 fps
i.	Video 1	30 fps	0.5x	15 fps
j.	Video 2	60 fps	0.5x	30 fps

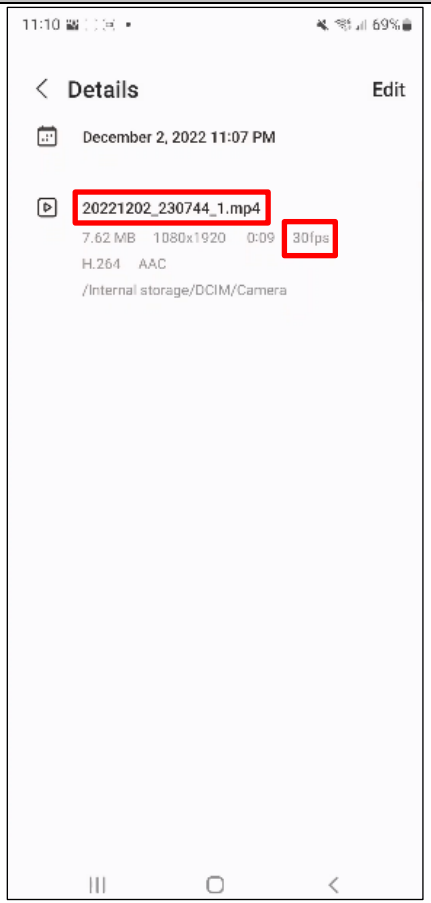
Source: Testing

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Converted Video 1 – Source: Testing  Converted Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products



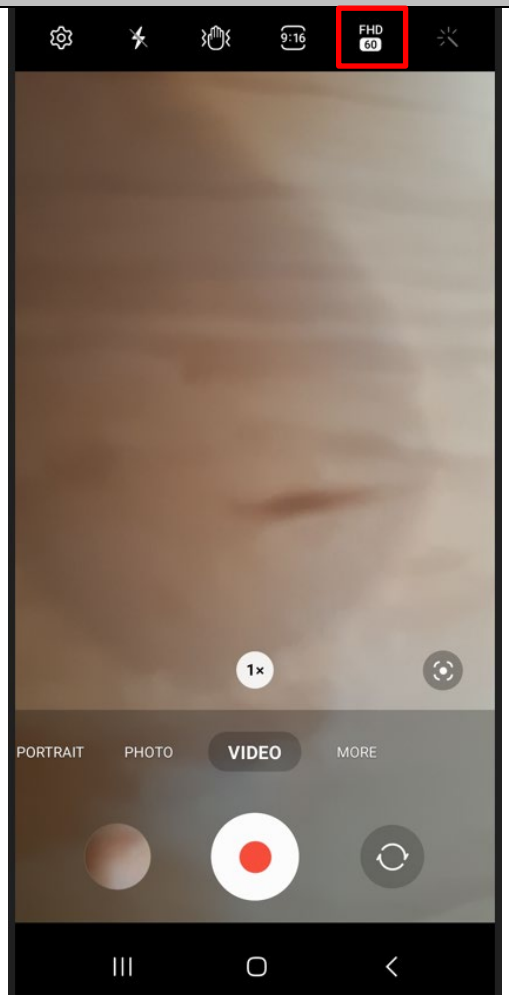
Converted Video 2 – Source: Testing

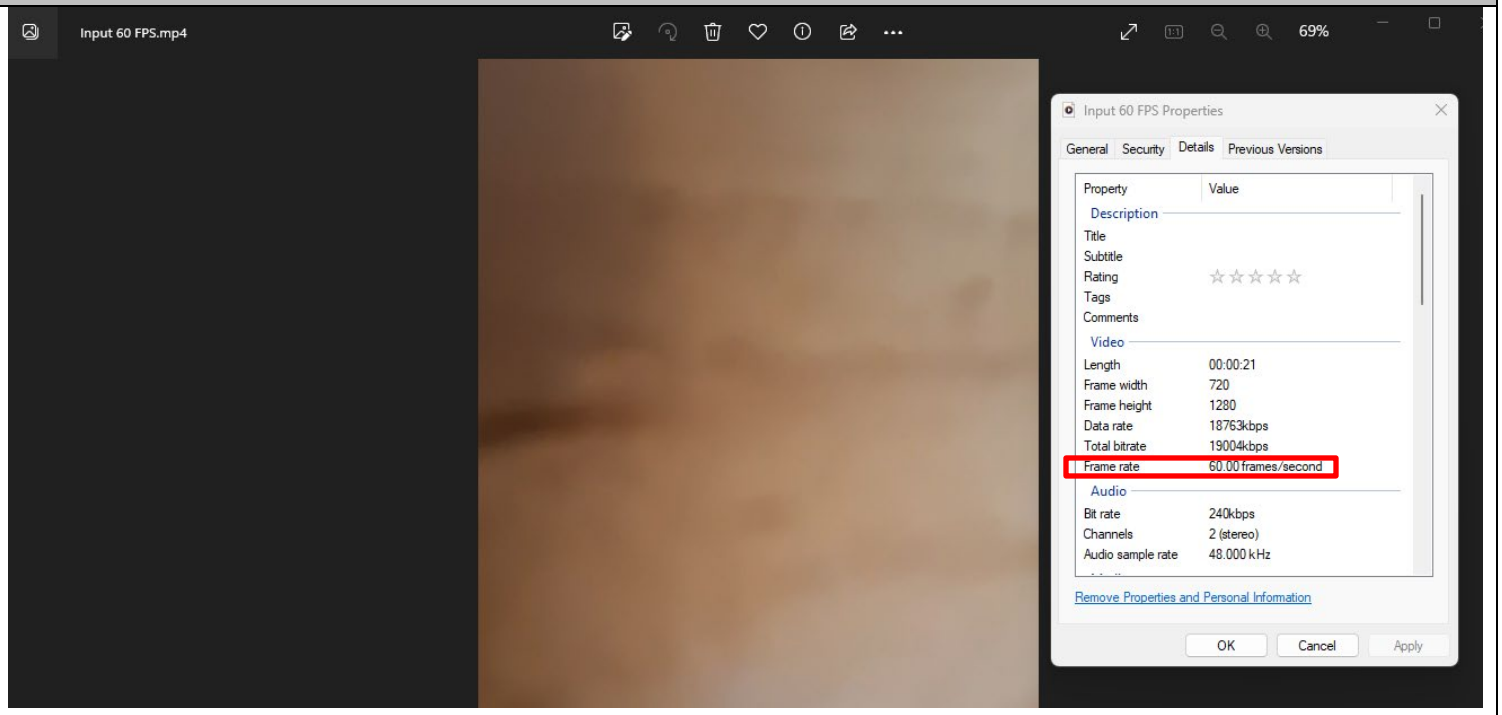


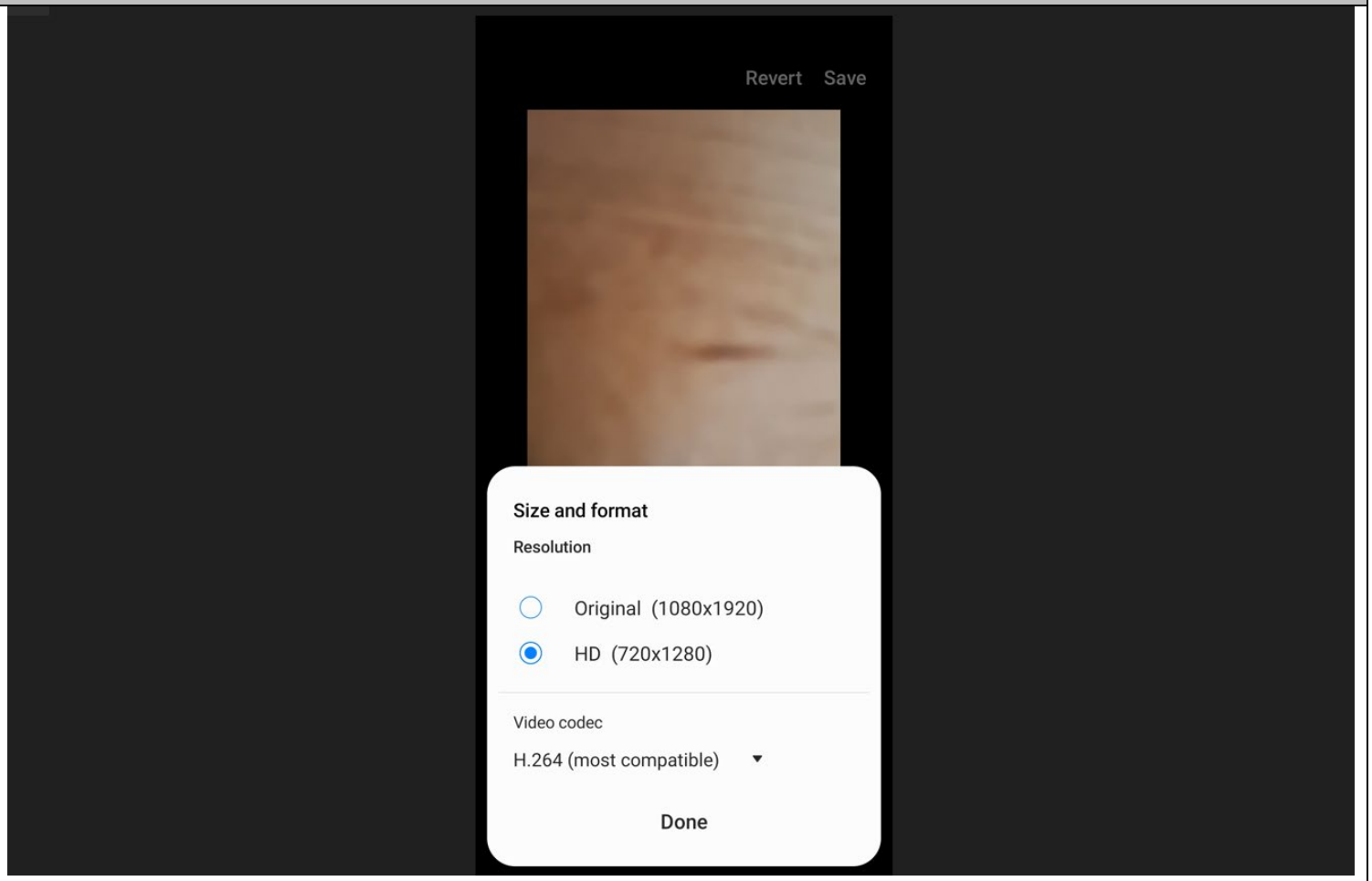
Converted Video 2 Details

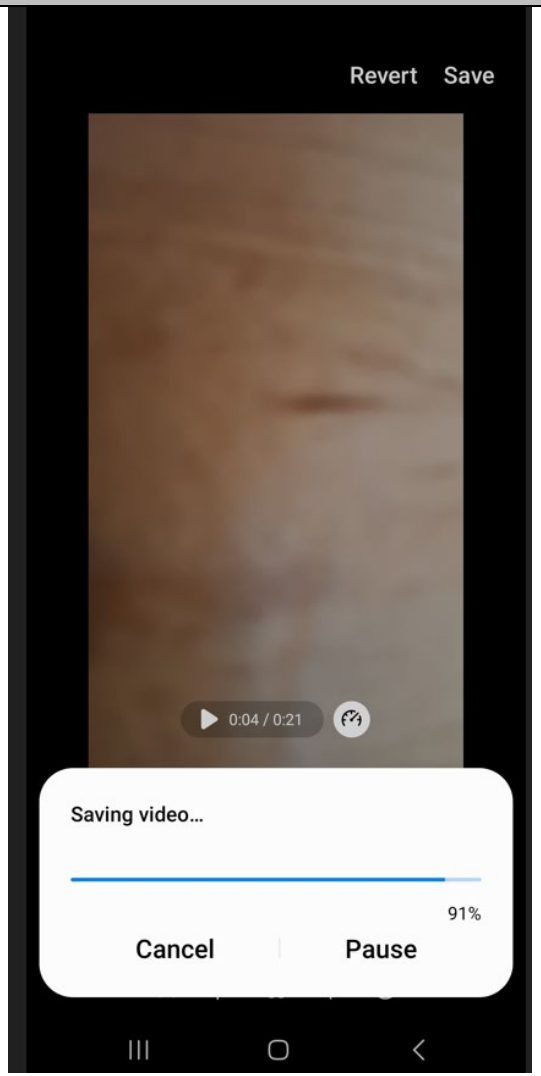
For example, using Super Slow-mo, a frame rate conversion unit *(i.e.*, "frame rate conversion tech") is configured to change the frame rate of a recorded .5 second video captured at 480 fps into a 16 second video captured at 960 fps. That is 1/32 the playback speed but double the frame rate.

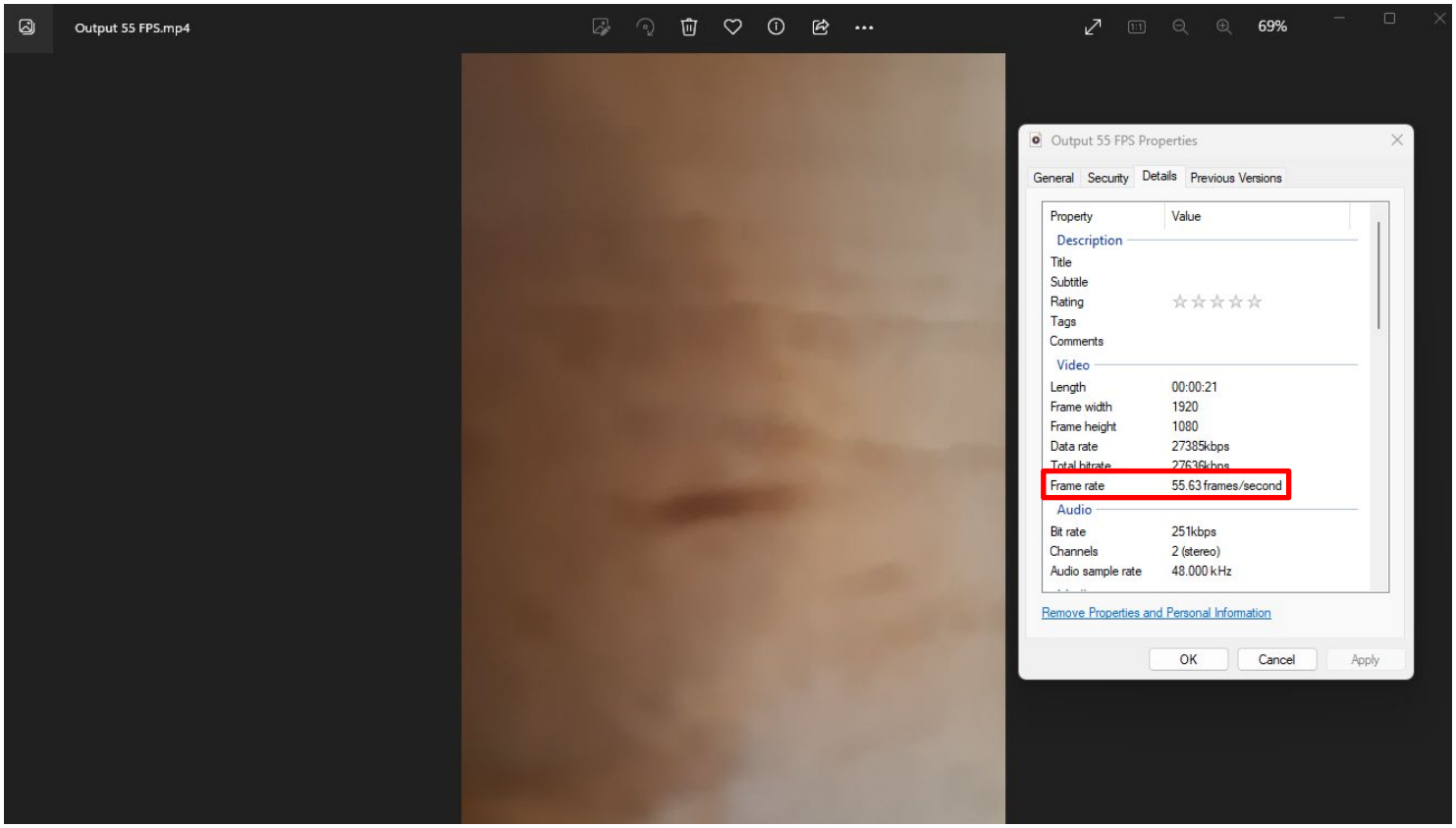
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	*Super Slow-mo only supports HD resolution. On Galaxy S22, S22+ and S22 Ultra, users can record approximately 0.5 seconds of video captured at 960 fps with approximately 16 seconds of playback. On Galaxy S22 Series, users can record approximately 0.5 second of video captured at 480 fps and digitally enhance the video to 960 fps with approximately 16 seconds of playback. Playback time can be edited in Super Slow-mo player. https://www.samsung.com/global/galaxy/galaxy-s22-ultra/specs/ As a little refresher, Slow Motion lets you play a recorded video up to 8 times slower than normal and its amped-up version, Super Slow-mo, plays clips 32 times slower than normal videos. This makes videos appear more artistic and could also be used for adding emphasis. The regular and Plus S22 variants can record 960 fps natively, whereas the S22 Ultra can only record 480fps super slow-motion videos, and then uses the frame rate conversion tech to reach 960fps. https://www.phonearena.com/news/samsung-galaxy-s22-ultra-480fps-slow-motion-video-not-supported_id139224 A frame rate can also be converted by adjusting the output frame rate through resolution settings. In the example below, the input frame rate chosen by selecting FHD, is 60 fps.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing

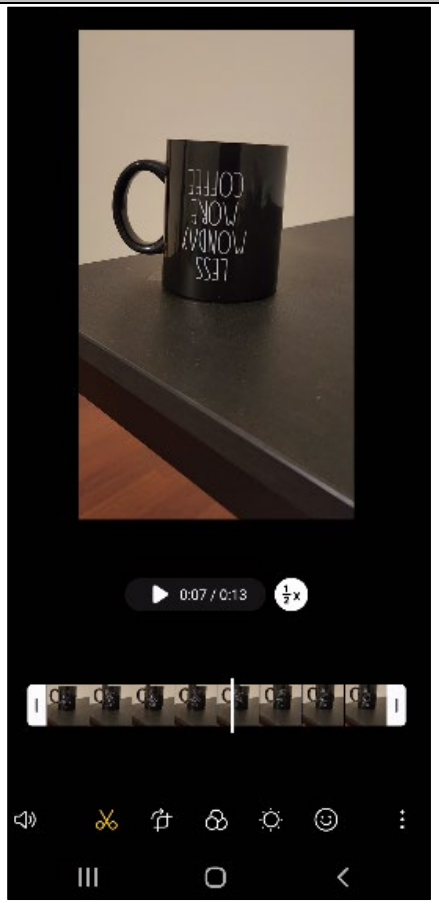
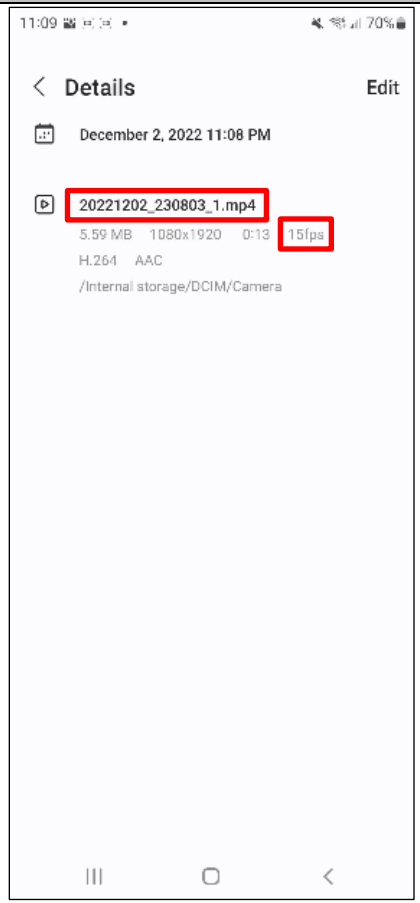
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing By changing the resolution from "Original, (1080x1920)" to "HD (729x1280)," and then saving the resulting video, the output frame rate is converted from the input frame rate of 60 fps to 55.63 fps.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing

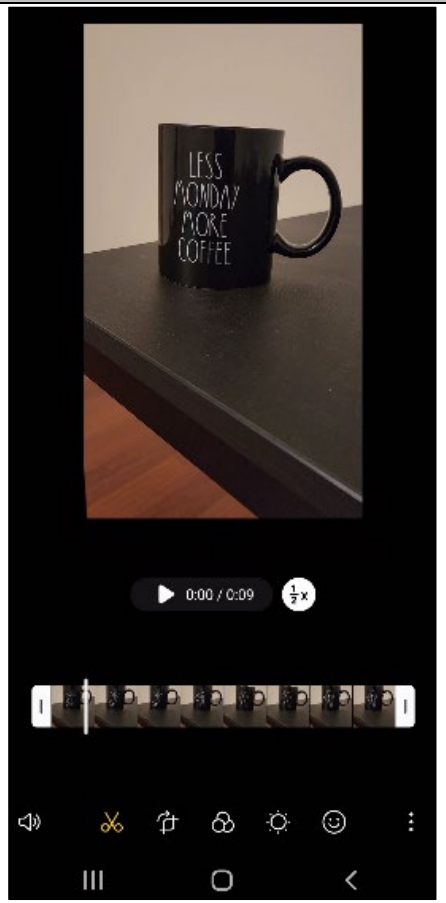
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Source: Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

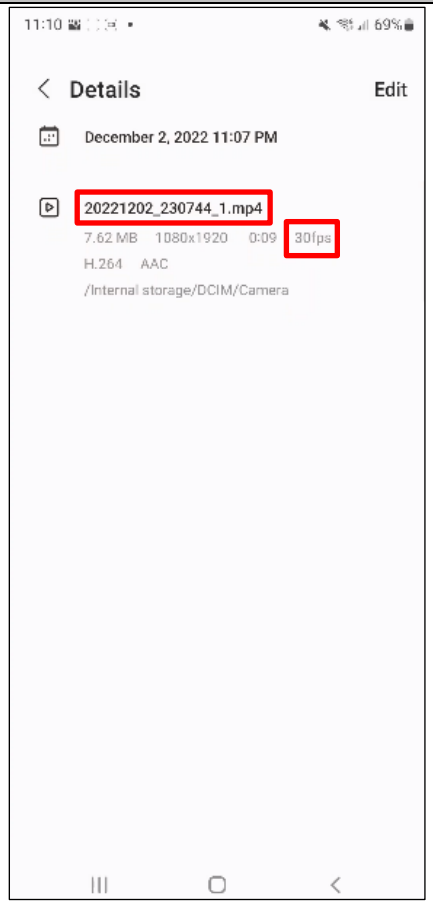
Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[6G] outputting the first and second moving image data whose frame rates have been changed.	Users of the Samsung Accused Products perform: outputting the first and second moving image data whose frame rates have been changed. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung Gallery in combination with the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone outputs (*e.g.*, creates a new .mp4 video file) the first and second moving image data whose frame rates have been converted (*e.g.*, the frame rate of Video 1 was converted from 30 frames per second to 15 frames per second, the frame rate of Video 2 was converted from 60 frames per second to 30 frames per second) by the frame rate conversion unit (*e.g.*, the Samsung Video Editor in the Samsung Galaxy S22 Ultra smartphone).Both have a playback speed of 0.5x.

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Converted Video 1 – Source: Testing  Converted Video 1 Details

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products



Converted Video 2 – Source: Testing



Converted Video 2 Details

Example	Input Frame Rate	Playback Speed	Changed Frame Rate

Claim – 9,445,041	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	<table><tr><td>Video 1</td><td>30 fps</td><td>0.5x</td><td>15 fps</td></tr><tr><td>Video 2</td><td>60 fps</td><td>0.5x</td><td>30 fps</td></tr></table> Source: Testing Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claim 1 (the "Asserted Claim") of U.S. Patent No. 9,986,303 (the "'303 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung 8K Neo QLED QN800A television, depicted in this chart, is believed to be representative of other televisions and display products, including at least the AU7000, AU8000, AU9000, BU8000, BU8500, Q60A, Q60B, Q60C, Q700T, Q70A, Q70B, Q70C, Q800T, Q80A, Q80B, Q80C, Q900TS, Q90T, Q950TS, Q95T, QN700A, QN700B, QN700C, QN800A, QN800B, QN800C, QN85A, QN85B, QN85C, QN900A, QN900B, QN900C, QN90A, QN90B, QN91A, QN95A, QN95B, QN95C, S90C, S95B, S95C, The Frame 2021, The Frame 2022, The Serif 2022, Q95T, Q90T, Q80T, Q70T, Q60T, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000, LS01T, LS03A, LS03T, LS05T, LST7, QP75A-8K, LSP9T, HBU8000, HQ60B, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra, Galaxy Book Pro, Galaxy Book Pro 360, Galaxy Book2 Pro, Galaxy Book2 Pro 360, Galaxy Book2 Business 14', Galaxy Book Odyssey, Galaxy Book3 Ultra, Galaxy Book3 Pro, Galaxy Book3 Pro 360, and Galaxy Book3 360 models for the purposes of infringement of the '303 Patent. Based on publicly available information, ACT believes that the Samsung 8K Neo QLED QN800A television, charted herein, and other televisions/display products share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the listed televisions/display products all purport to support the AV1 codec. Should discovery yield documents or information demonstrating that Defendants sell other infringing products which are not represented by the Samsung 8K Neo QLED QN800A television, ACT reserves its right to amend its infringement contentions to include a chart mapping the claims of the '303 Patent to the structure and function of the newly-discovered products.

Samsung directly infringes the Asserted Claim by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

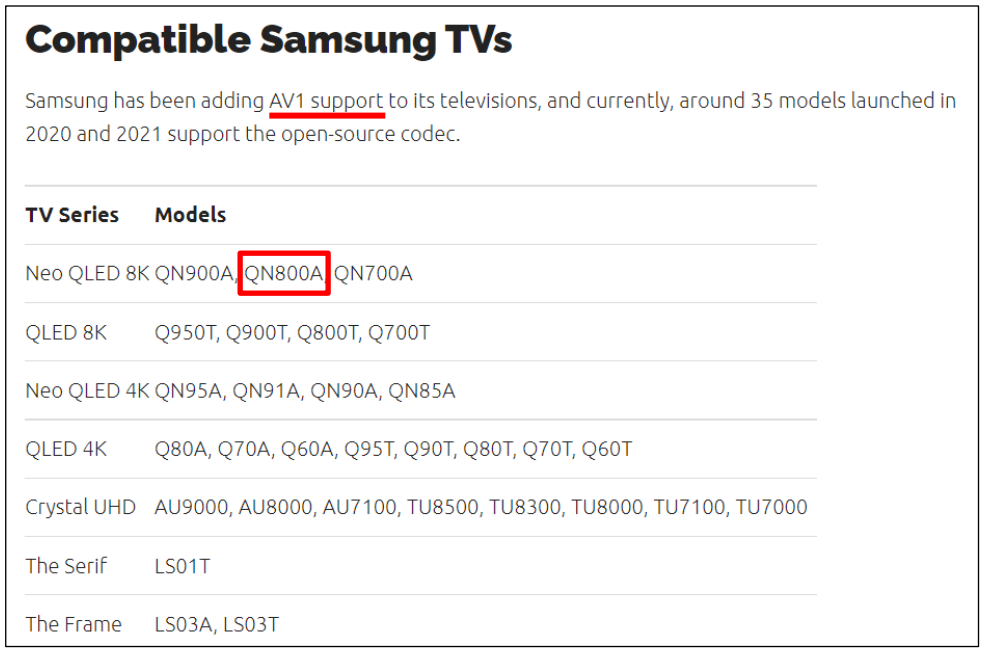
Samsung indirectly infringes the Asserted Claim in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through its operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claim; and (ii) marketing and

advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers, including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claim, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claim but has taken deliberate actions to avoid learning these facts.

Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

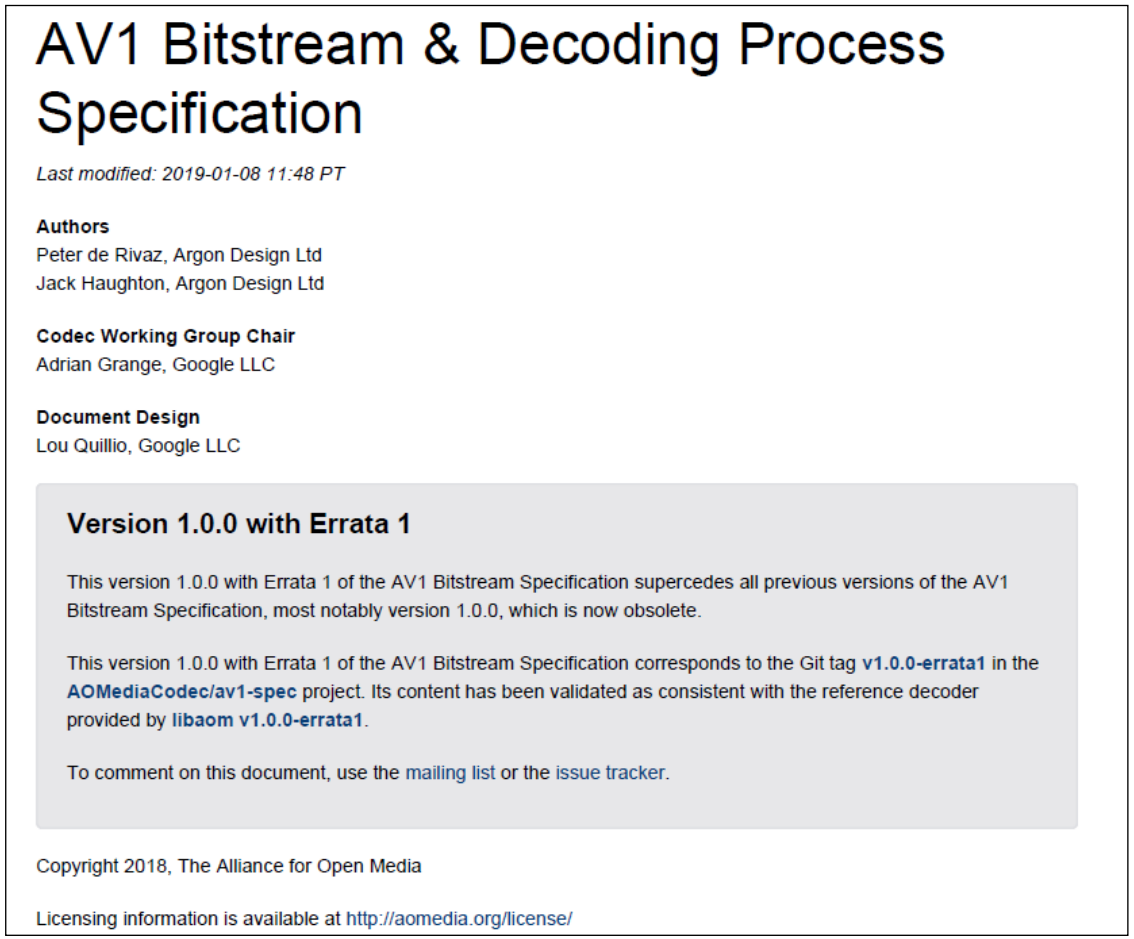
ACT does not concede that any claims of the '303 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1P] A video image coding data receiver comprising:	The Samsung Accused Products comprise: A video image coding data receiver. Samsung infringes directly, and/or indirectly by providing a video image coding data receiver, and encouraging others, including customers and end users, to use the video image coding data receiver. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a video image coding data receiver that supports decoding of the AV1 codec.  *See e.g.*, https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products

The AV1 codec is used for video coding/decoding.



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

See e.g., https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products

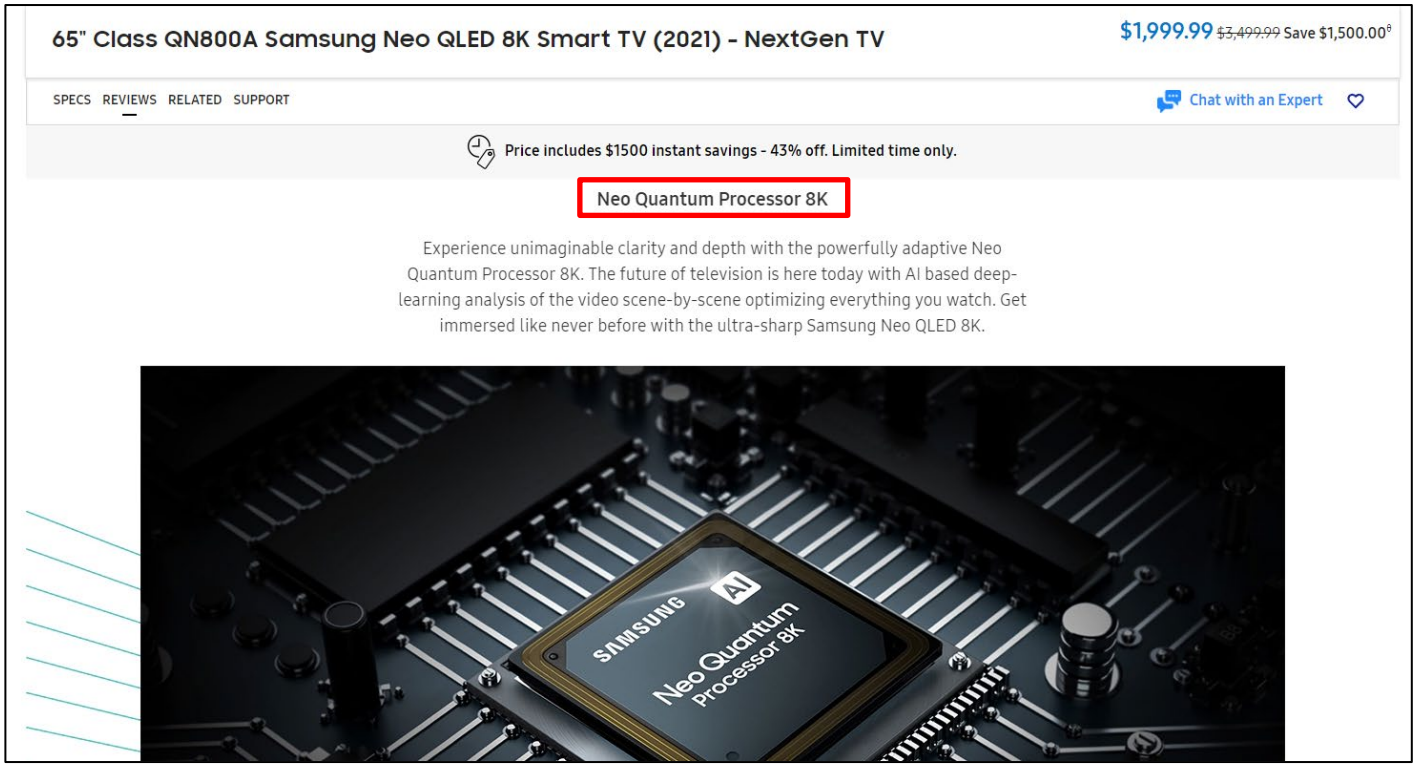
The Samsung 8K Neo QLED QN800A television supports the AV1 codec.

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

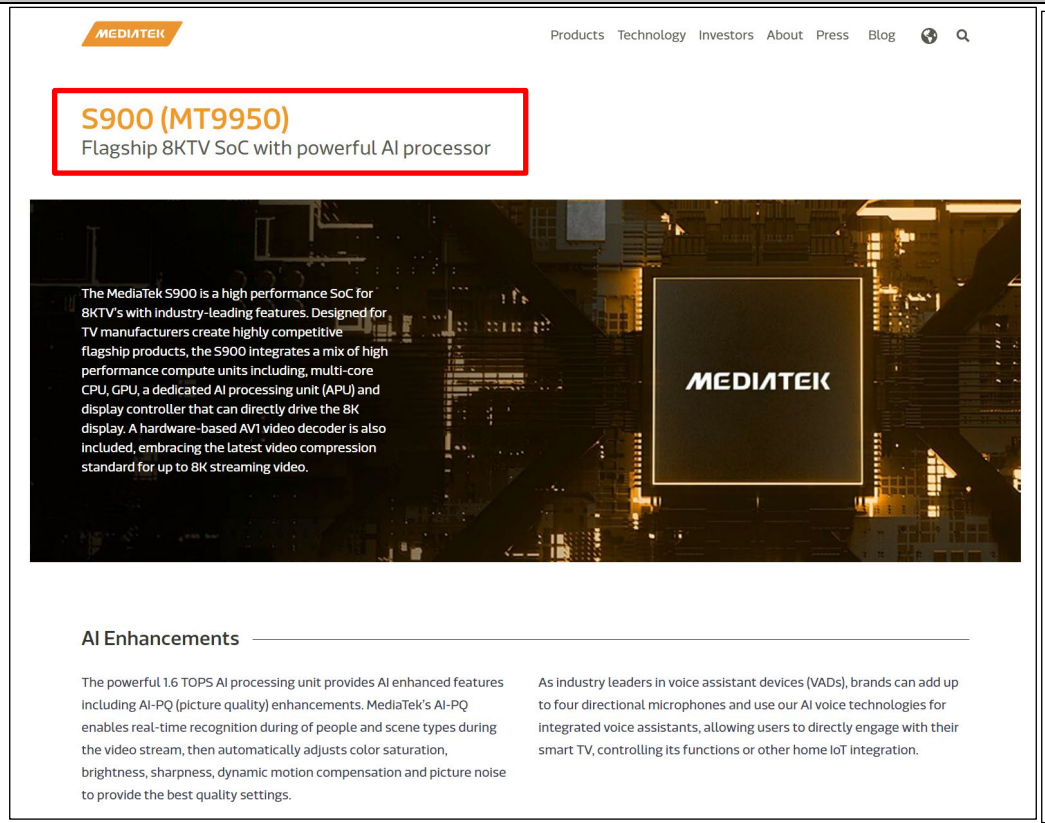
https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-

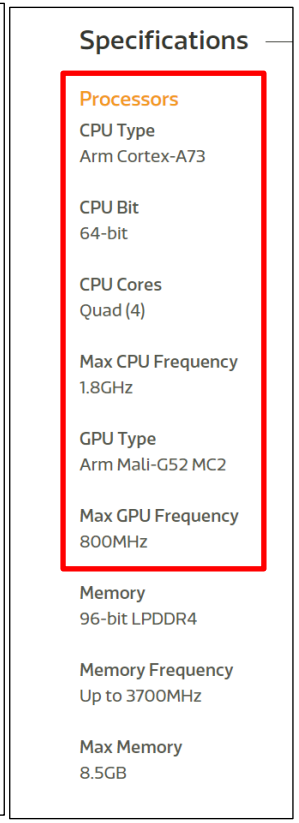
Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1A] a processor; and	The Samsung Accused Products comprise a processor. Samsung infringes directly, and/or indirectly by providing a processor, and encouraging others, including customers and end users, to use the processor. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices with the processor. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a Neo Quantum Processor 8K.

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://www.samsung.com/us/televisions-home-theater/tvs/samsung-neo-qled-8k/65-qn800a-neo-qled-8k-smart-tv-2021-qn65qn800afxza/

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The Neo Quantum Processor 8K is a MediaTek S900 (MT9950) processor. **Neo Quantum processor 8K specifications** The Neo Quantum processor 8K is a MediaTek S900 (MT9950) processor. This processor features support for 8K screen resolution. The processor is built on Arm Cortex-A73 cores with a frequency of 1.8GHz, 4 cores, a graphics processor Arm Mali-G52 MC2 0.8Ghz. With the 96-bit LPDDR4 memory, the processor supports a maximum of 8.5GB of memory, the memory runs at 3.7GHz. https://www.tab-tv.com/neo-quantum-processor-samsung-tv-specifications/

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products



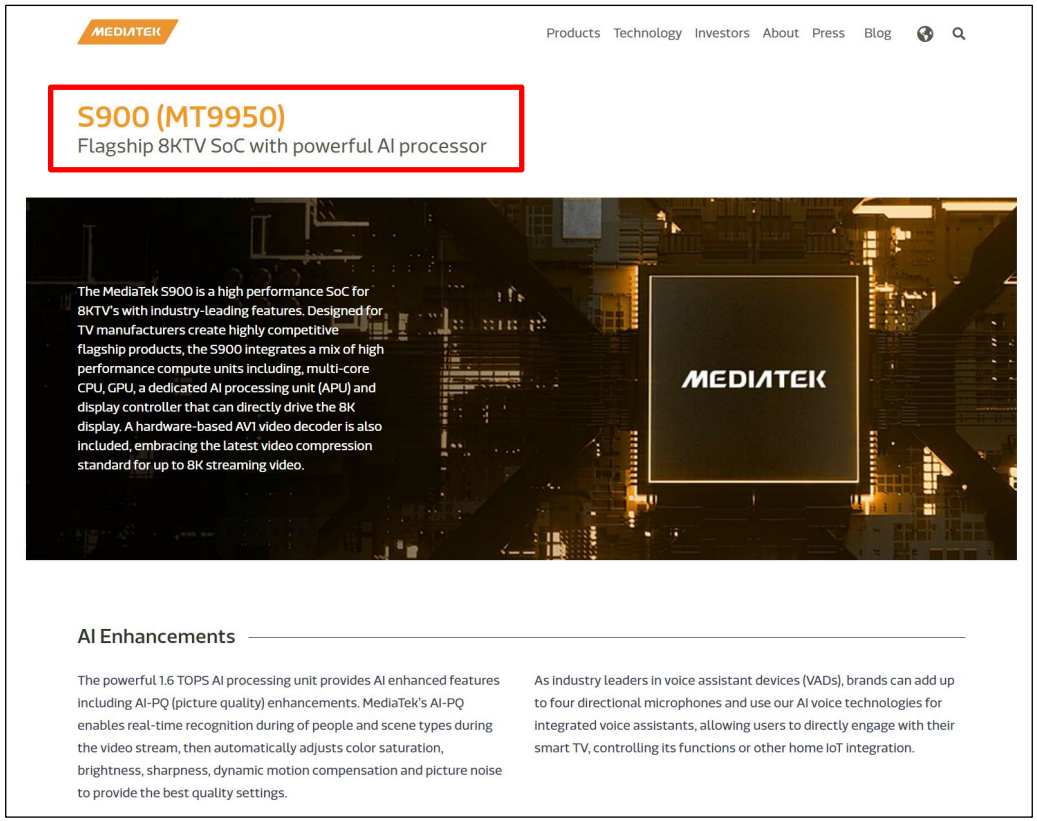


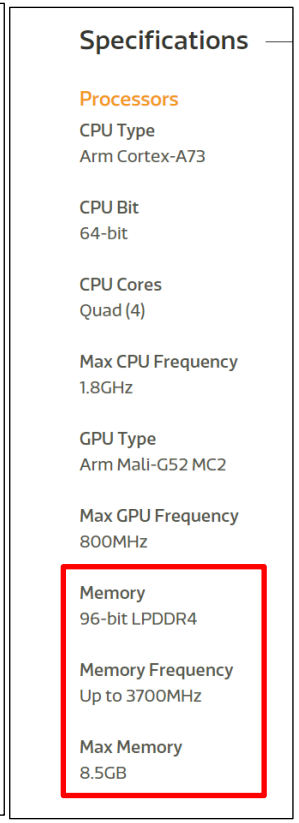
https://www.mediatek.com/products/digital-tv/s900-mt9950

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1B] a memory unit having instructions stored which, when executed by the processor, cause the processor to perform operations comprising:	The Samsung Accused Products comprise a memory unit having instructions stored which, when executed by the processor, cause the processor to process video image coding data. Samsung infringes directly, and/or indirectly by providing a memory unit having instructions stored which, when executed by the processor, cause the processor to process video image coding data, and encouraging others, including customers and end users, use the memory unit. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices with the memory unit having instructions stored which, when executed by the processor, cause the processor to process video image coding data. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises the MediaTek S900 (MT9950) processor which comprises a memory unit having up to 8.5GB of 96-bit LPDDR4 memory. **Neo Quantum processor 8K specifications** The Neo Quantum processor 8K is a MediaTek S900 (MT9950) processor. This processor features support for 8K screen resolution. The processor is built on Arm Cortex-A73 cores with a frequency of 1.8GHz, 4 cores, a graphics processor Arm Mali-G52 MC2 0.8Ghz. With the 96-bit LPDDR4 memory, the processor supports a maximum of 8.5GB of memory, the memory runs at 3.7GHz. https://www.tab-tv.com/neo-quantum-processor-samsung-tv-specifications/

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products





https://www.mediatek.com/products/digital-tv/s900-mt9950

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[1C] receiving basic video image coding data; decoding the received basic video image coding data so as to reproduce a video image;	The Samsung Accused Products receive basic video image coding data and decode the received basic video image coding data so as to reproduce a video image. Samsung infringes directly, and/or indirectly by receiving basic video image coding data and decoding the received basic video image coding data so as to reproduce a video image, and encouraging others, including customers and end users, to receive basic video image coding data and decode the received basic video image coding data so as to reproduce a video image. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices that receive basic video image coding data and decode the received basic video image coding data so as to reproduce a video image. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder in the Samsung 8K Neo QLED QN800A receives basic video image coding data (*e.g.*, a stream of video data) to be processed in a predictive-transform coding scheme. The Accused Products decode this basic video data to reproduce a video image (*i.e.*, "reconstruct[ed] frame" or "improve the perceptual quality of the displayed frames"). **Decoding process** **AV1 decoding process at higher level** The decoding process of the AV1 decoder model ties together operations on the smoothing buffer and the decoder frame buffer. In particular, the decoder model determines when a frame decoding starts and the frame bits are removed from the smoothing buffer, which instantaneously decreases fullness of the smoothing buffer by the corresponding amount. The decoder model also calculates when the decoding finishes and the decoded frame is added to the frame buffers. It also determines when the frame is output for the display and removed from the buffer.

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The AV1 codec supports input video signals in the 4:0:0 (monochrome), 4:2:0, 4:2:2, and 4:4:4 formats. The allowed pixel representations are 8, 10, and 12 bit. The AV1 codec operates on pixel blocks. Each pixel block is processed in a predictive-transform coding scheme, where the prediction comes from either intraframe reference pixels, interframe motion compensation, or some combinations of the two. The residuals undergo a 2-D unitary transform to further remove the spatial correlations, and the transform coefficients are quantized. Both the prediction syntax elements and the quantized transform coefficient indexes are then entropy coded using arithmetic coding. There are three optional in-loop postprocessing filter stages to enhance the quality of the reconstructed frame for reference by subsequent coded frames. A normative film grain synthesis unit is also available to improve the perceptual quality of the displayed frames. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products

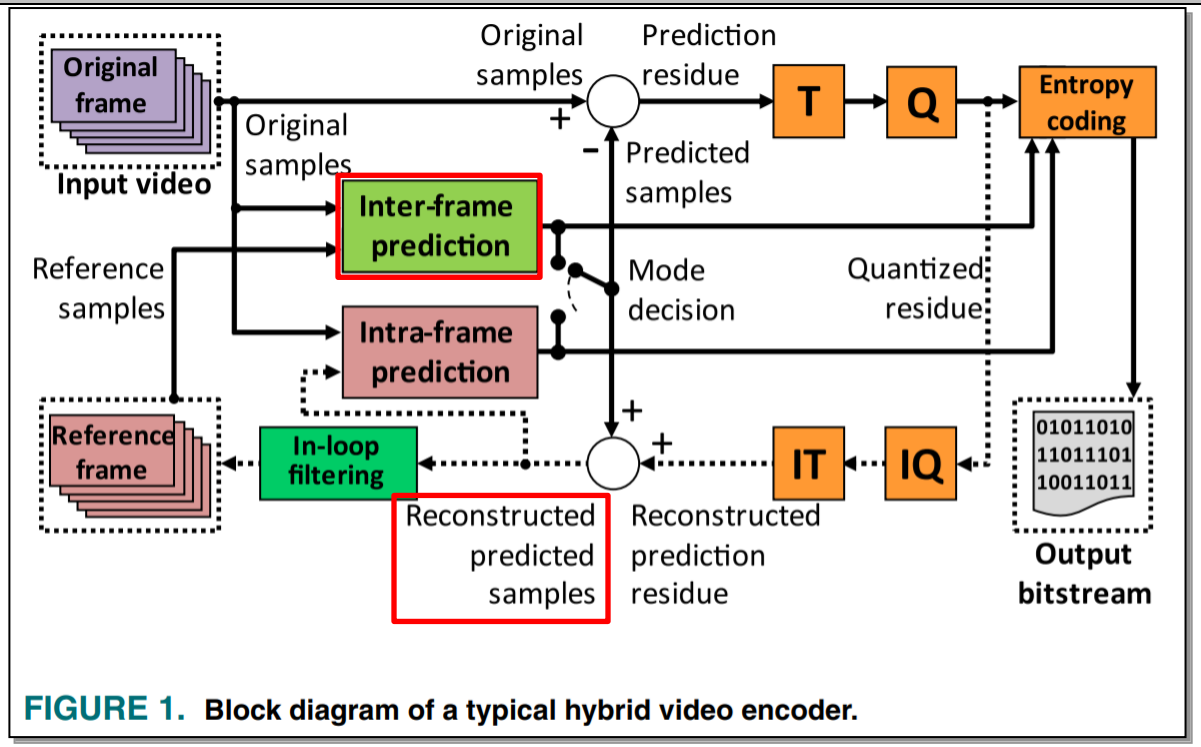
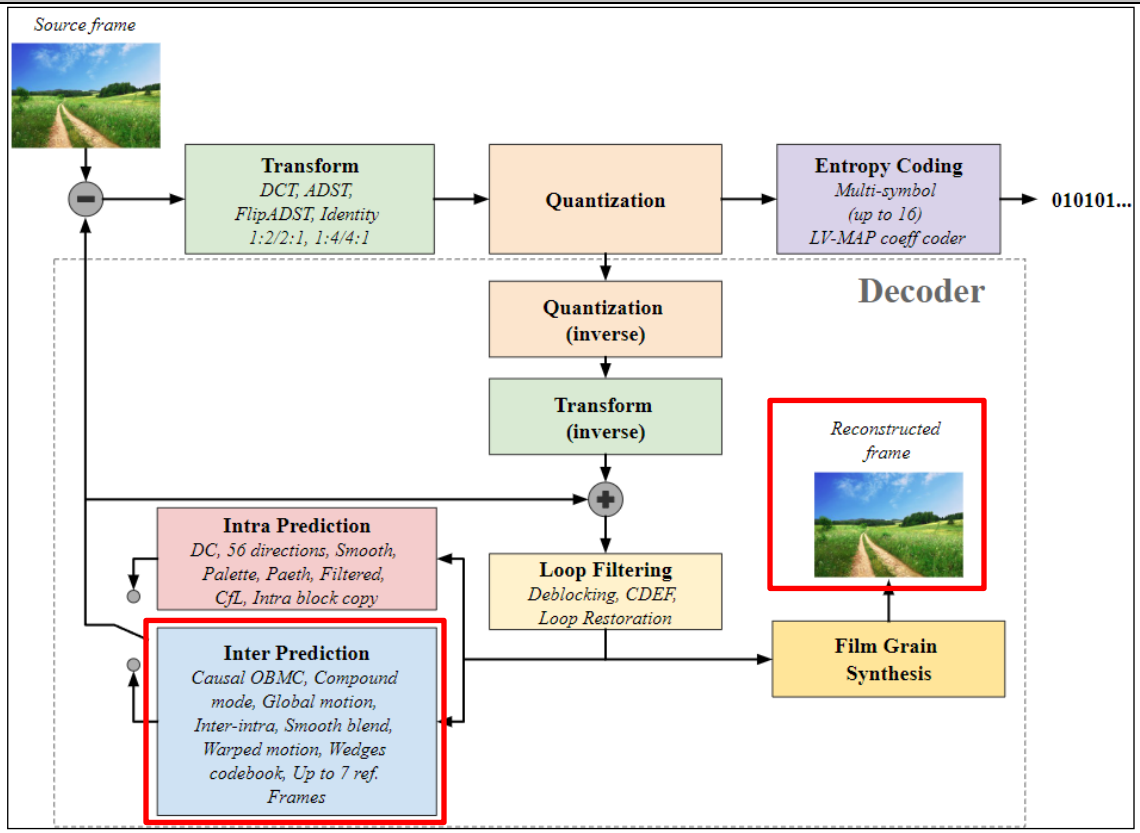


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=

Claim 1 – 9,986,303	Exemplary Supporting Evidence Regarding Samsung's Accused Products



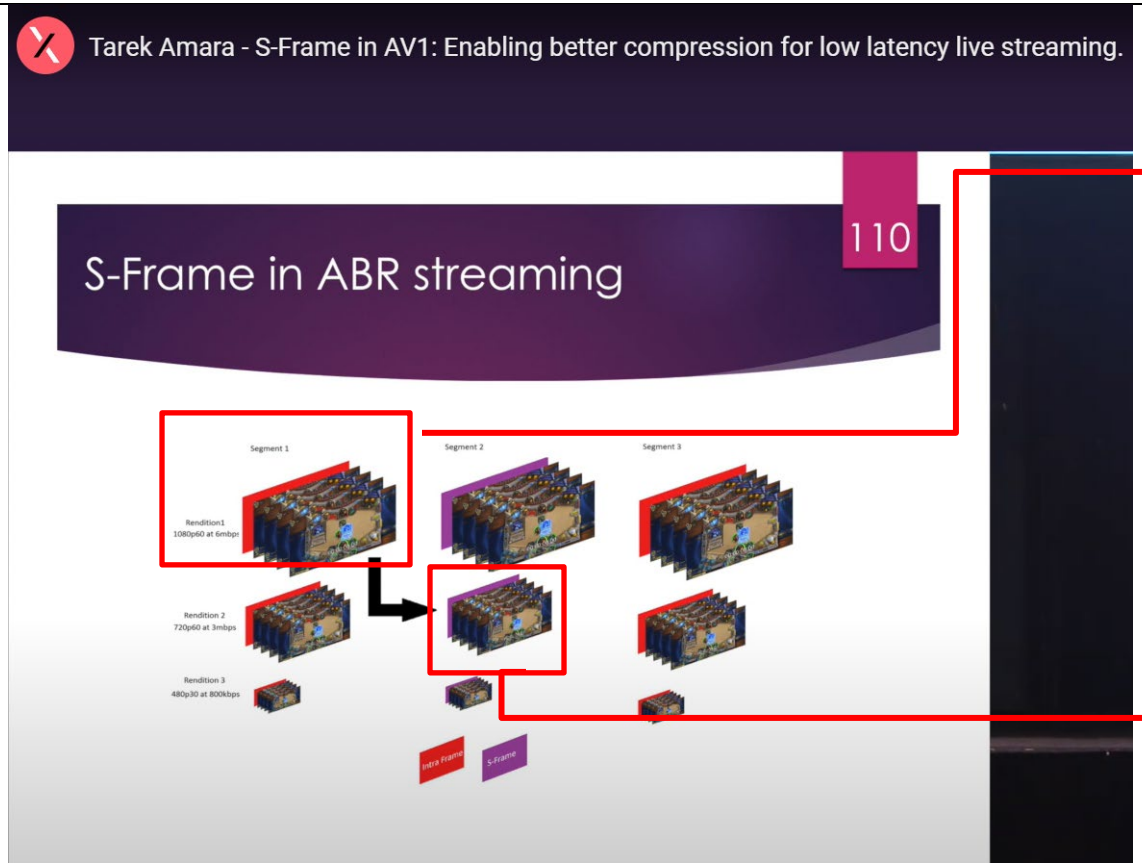
https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

[1D] receiving supplementary video image coding data including a supplementary hierarchical picture whose coding order and display order are earlier by a factor of a group of pictures including an intra coded picture and a plurality of inter prediction coded pictures than those of a basic hierarchical picture included in the basic video image coding data, a basic hierarchy and a supplementary hierarchy being set in units of	The Samsung Accused Products receive supplementary video image coding data including a supplementary hierarchical picture whose coding order and display order are earlier by a factor of a group of pictures including an intra coded picture and a plurality of inter prediction coded pictures than those of a basic hierarchical picture included in the basic video image coding data, a basic hierarchy and a supplementary hierarchy being set in units of the group of pictures. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices with the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder in the Samsung 8K Neo QLED QN800A receives both basic video image coding data (*i.e.*, a sequence of pictures or frames received at a certain bitrate or bandwidth) and supplementary video image coding data (*i.e.*, a sequence of pictures or frames with a different bitrate or bandwidth than the basic video image coding data). The supplementary video image coding data includes a supplementary hierarchical picture (*i.e.*, switch frame). The switch frame has an earlier coding order and display order than those of a basic hierarchical picture included in the basic video coding data because the switch frame is based on previously decoded buffered reference frames. Additionally, "small chunks (say of 1-second duration), each start[] with a switch frame." Each "small chunk" is a group of pictures (GOP). There may be one or more switch frame chunks. A GOP is a grouping of a series of frames into sequences. **Switch Frame** An inter frame that can be used as a point to switch between sequences. Switch frames overwrite all the reference frames without forcing the use of intra coding. The intention is to allow a streaming use case where videos can be encoded in small chunks (say of 1 second duration), each starting with a switch frame. If the available bandwidth drops, the server can start sending chunks from a lower bitrate encoding instead. When this happens the inter prediction uses the existing higher quality reference frames to decode the switch frame. This approach allows a bitrate switch without the cost of a full key frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5

the group of pictures;	For example, when bandwidth drops, the decoder uses existing higher-quality reference frames to decode the switch frame. When the decoder starts decoding the lower bitrate rendition starting with a switch frame, the higher bitrate frame decoded in the previous segment is still in the memory. When a lower bitrate rendition starting with a switch frame is received, the higher resolution GOPs/segments that were downloaded as a part of the adaptive streaming cache will not be decoded, since decoding them would cause the decoder buffer to flush and lose the reference frame needed by switch frame. Instead, the lower bitrate stream decoding commences. The starting point of the lower bitrate switch point will be one or more GOPs earlier than the higher bitrate rendition GOPs that have already been downloaded to the cache (*i.e.*, the download is in units of GOPs).

603



https://thebroadcastknowledge.com/2020/04/15/video-s-frame-in-av1-enabling-better-compression-for-low-latency-live-streaming/ ("So let's take this small example. So someone tuned into segment 1 which is 6mbps and then he is going to switch segment to a friend at segment 2. We hope that segment we can started with our effective or not defined yet s-frame which is an interframe. So to decode that purple frame from the middle segment, it's an interframe so what we require we said we require the elementary stream which is already there. We also need the reference frames. In order to have access to the reference frames, we should have decoded these frames so these are the real reference frames to this one. However, comparing those two segments to each other, we do realize they represent the same video. They represent same

data. So here comes the idea – why don't we just choose those physical reference frames which already in the buffer because your player or your decoder have already decoded all these frames and instead of finding a way to use this will be a replacement of reference frames that should have come from this rendition.").

temporal_group_size indicates the number of pictures in a temporal picture group. If the temporal_group_size is greater than 0, then the scalability structure data allows the inter-picture temporal dependency structure of the coded video sequence to be specified. If the temporal_group_size is greater than 0, then for temporal_group_size pictures in the temporal group, each picture's temporal layer id (temporal_id), switch up points (temporal_group_temporal_switching_up_point_flag and temporal_group_spatial_switching_up_point_flag), and the reference picture indices (temporal_group_ref_pic_diff) are specified.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 137

Additionally, the group of pictures includes both an intra coded picture (*i.e.*, intra frame or intra_only_frame) and a plurality of inter prediction coded pictures (*i.e.*, inter frames or inter_frames).

frame_type specifies the type of the frame:

:

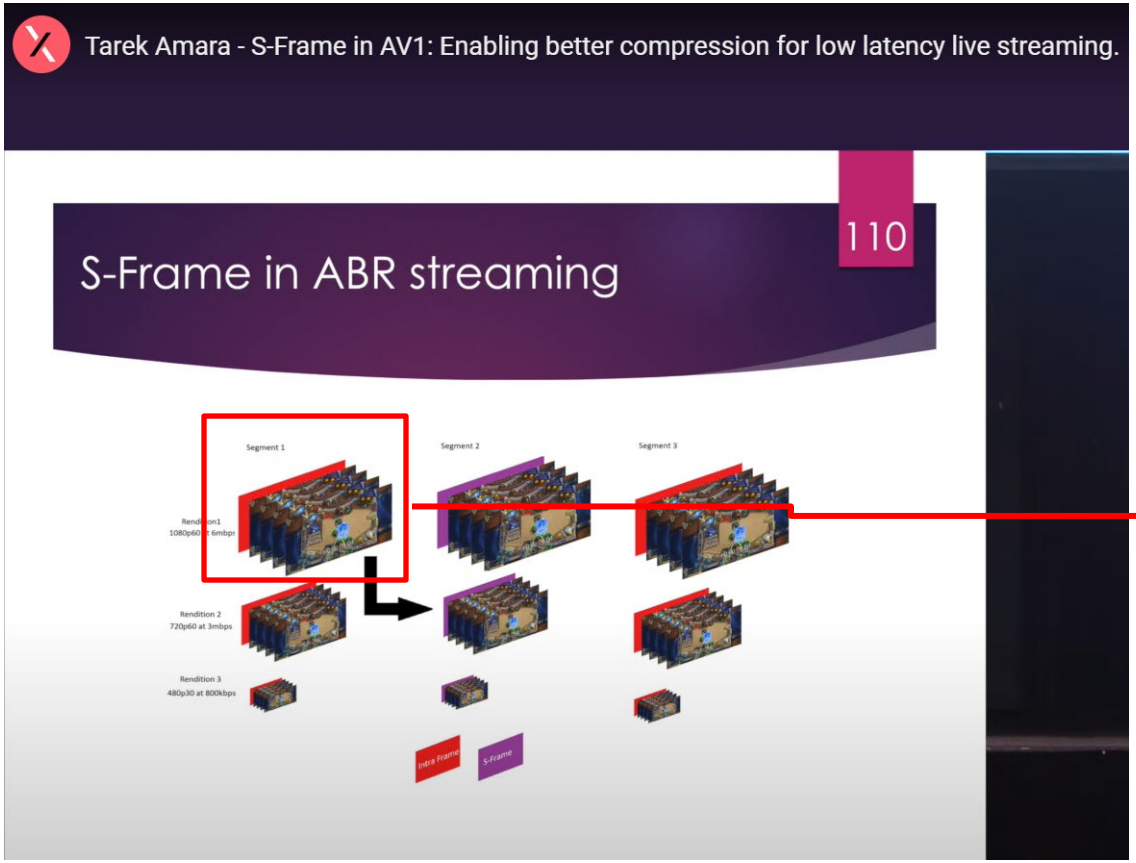
Name of frame_type
KEY_FRAME
INTER_FRAME
INTRA_ONLY_FRAME
SWITCH_FRAME

605

	https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 150. **Inter coding** Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. **Intra frame** A frame compressed using only intra prediction which can be independently decoded. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1E] acquiring basic video image coding data received before supplementary video image coding data that has been	The Samsung Accused Products acquire basic video image coding data received before supplementary video image coding data that has been received at the moment. Samsung infringes directly, and/or indirectly by providing a processor, and encouraging others, including customers and end users, use the processor. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices with the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder in the Samsung 8K Neo QLED QN800A acquires basic video image coding data (*i.e.*, a sequence of pictures or frames with a certain bitrate or bandwidth) received before supplementary video image coding data (*i.e.*, a sequence of pictures or frames with a different bitrate or bandwidth than the basic

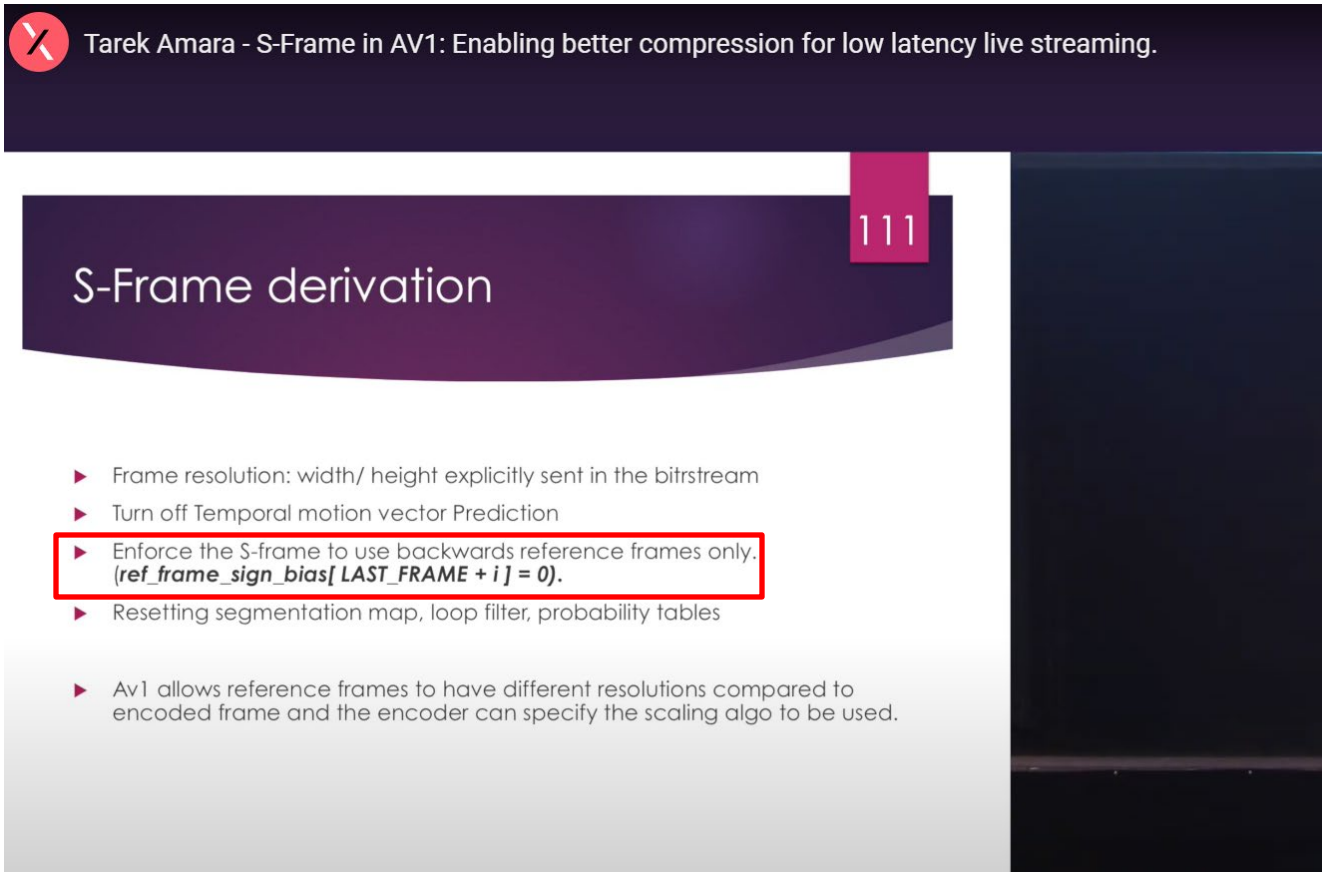
received at the moment; and	video image coding data). Because the supplementary hierarchical picture (*i.e.*, switch frame) is used as a point to switch between sequences of basic and supplementary hierarchical image coding data, there must be basic video image coding data received before the supplementary video image coding data received at the moment. For example, when bandwidth drops (*e.g.*, the supplementary video image coding data has a lower bandwidth or bit rate than the preceding basic video image coding data), inter prediction uses the existing higher quality reference frames from the preceding basic video image coding data to decode the switch frame, which is the supplementary hierarchical picture from the supplementary video image coding data. **Switch Frame** An inter frame that can be used as a point to switch between sequences. Switch frames overwrite all the reference frames without forcing the use of intra coding. The intention is to allow a streaming use case where videos can be encoded in small chunks (say of 1 second duration), each starting with a switch frame. If the available bandwidth drops, the server can start sending chunks from a lower bitrate encoding instead. When this happens the inter prediction uses the existing higher quality reference frames to decode the switch frame. This approach allows a bitrate switch without the cost of a full key frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5 **Inter coding** Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 3

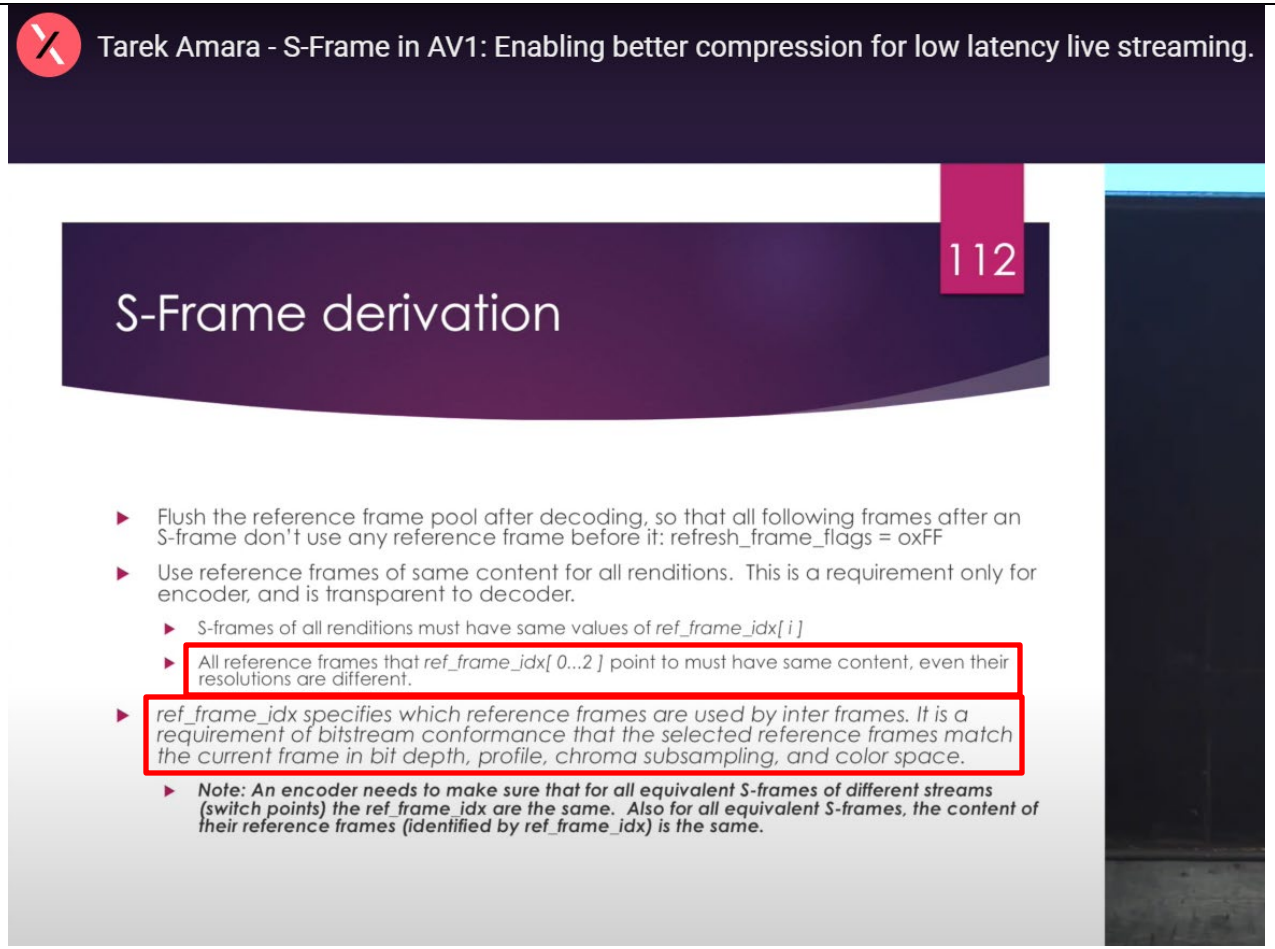
809



Acquired basic video image coding data received before supplementary video image coding data

https://thebroadcastknowledge.com/2020/04/15/video-s-frame-in-av1-enabling-better-compression-for-low-latency-live-streaming/


Tarek Amara - S-Frame in AV1: Enabling better compression for low latency live streaming.
111
S-Frame derivation
Frame resolution: width/ height explicitly sent in the bitrstream
Turn off Temporal motion vector Prediction
Enforce the S-frame to use backwards reference frames only. (ref_frame_sign_bias[LAST_FRAME + i] = 0).
Resetting segmentation map, loop filter, probability tables
Av1 allows reference frames to have different resolutions compared to encoded frame and the encoder can specify the scaling algo to be used.

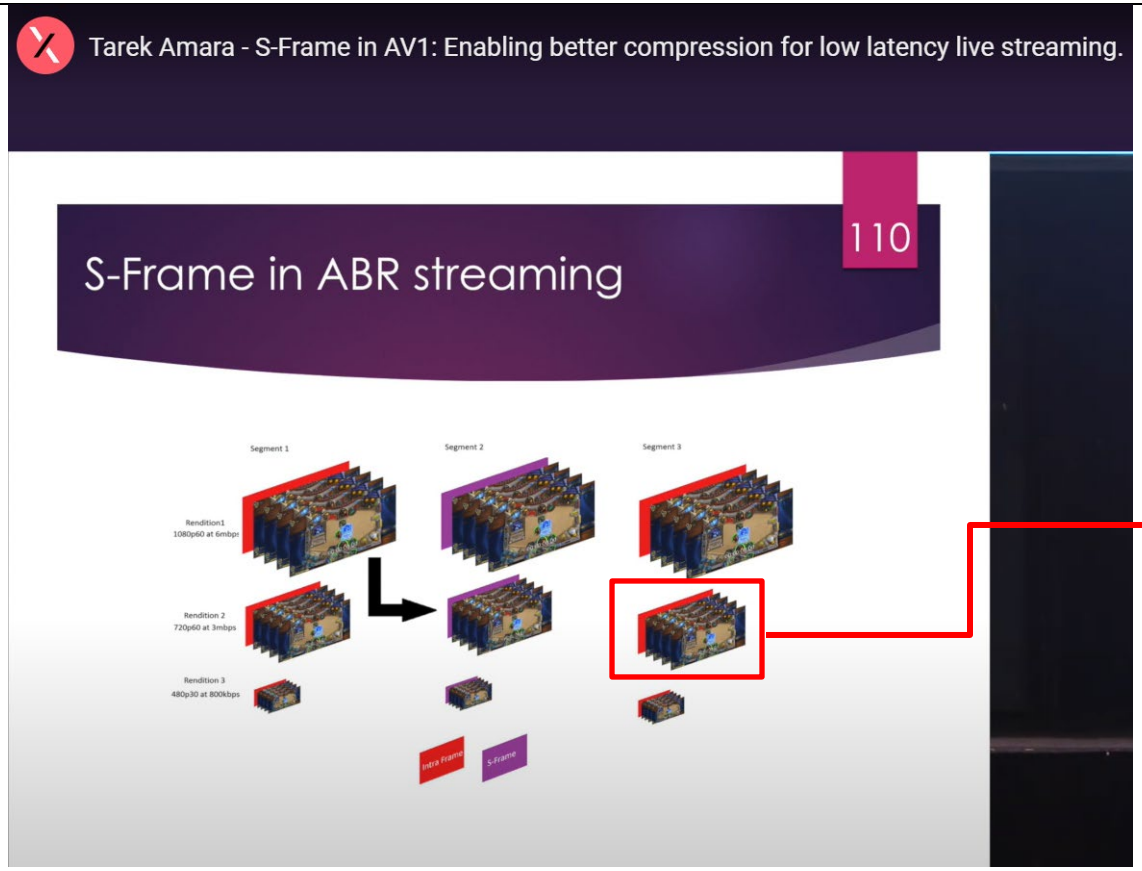


https://thebroadcastknowledge.com/2020/04/15/video-s-frame-in-av1-enabling-better-compression-for-low-latency-live-streaming/

	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[1F] reconstructing video image coding data from the basic video image coding data and the supplementary video image coding data.	The Samsung Accused Products reconstruct video image coding data from the basic video image coding data and the supplementary video image coding data. Samsung infringes directly, and/or indirectly by providing a processor, and encouraging others, including customers and end users, use the processor. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices with the A1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the AV1 decoder in the Samsung 8K Neo QLED QN800A uses inter prediction to reconstruct video image coding data from the basic video image coding data (*i.e.*, specifically previously decoded reference frames from a sequence of pictures or frames with a certain bitrate or bandwidth) and the supplementary video image coding data (*i.e.*, a sequence of pictures or frames with a different bitrate or bandwidth than the basic video image coding data). The reconstructed video image coding data is the output of the inter prediction process (*i.e.*, reconstructed samples in the current frame CurrFrame). **Switch Frame** An inter frame that can be used as a point to switch between sequences. Switch frames overwrite all the reference frames without forcing the use of intra coding. The intention is to allow a streaming use case where videos can be encoded in small chunks (say of 1 second duration), each starting with a switch frame. If the available bandwidth drops, the server can start sending chunks from a lower bitrate encoding instead. When this happens the inter prediction uses the existing higher quality reference frames to decode the switch frame. This approach allows a bitrate switch without the cost of a full key frame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 5

	Inter coding Coding one block or frame using inter prediction. **Inter frame** A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Inter prediction** The process of deriving the prediction value for the current frame using previously decoded frames. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 3

612



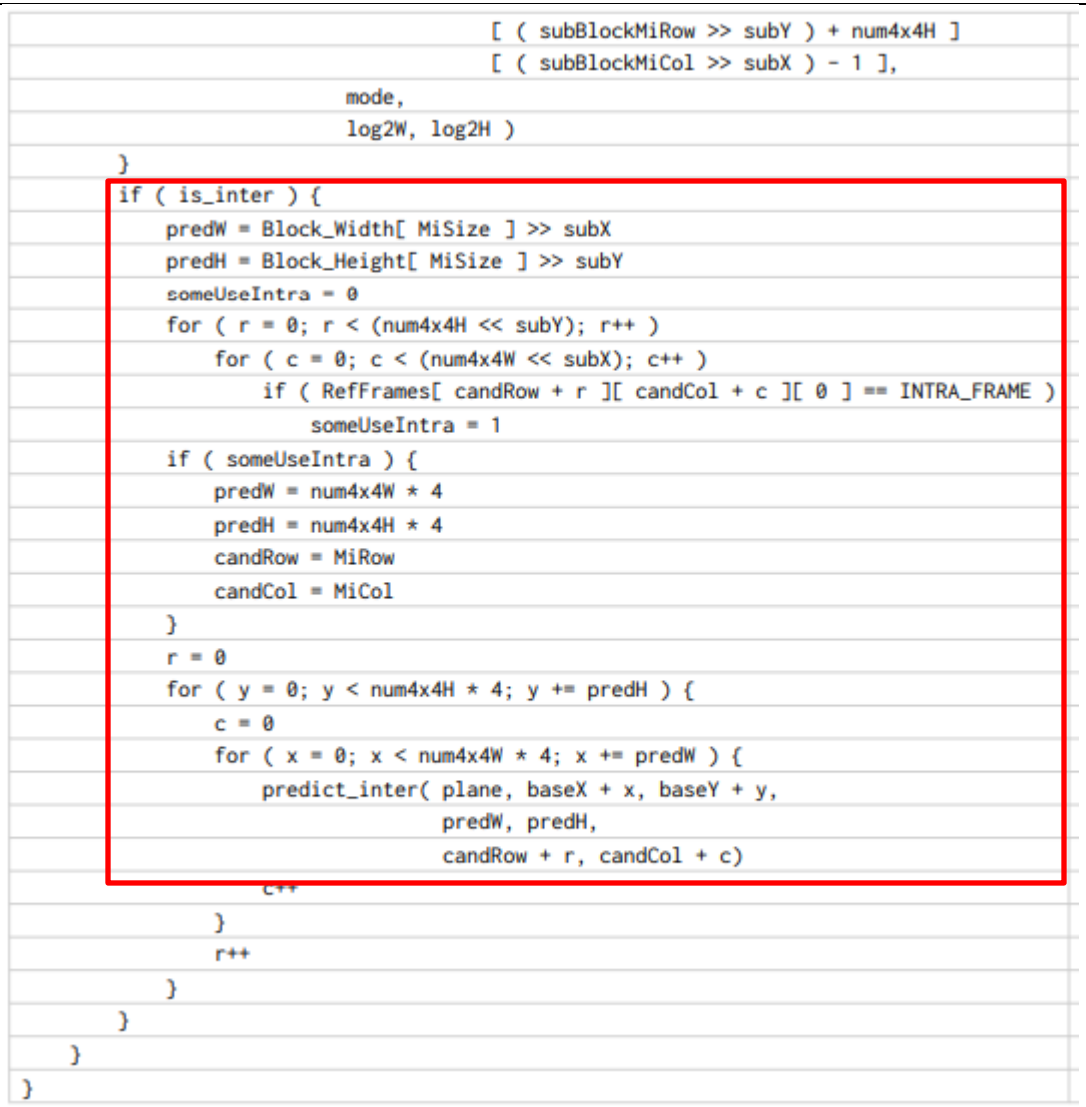
https://thebroadcastknowledge.com/2020/04/15/video-s-frame-in-av1-enabling-better-compression-for-low-latency-live-streaming/

Reconstructed video image coding data by combining the basic video image coding data at higher bandwidth (including reference frames for s-frame) and supplementary video image coding data at lower bandwidth (including s-frame itself)

	6.10.31. Compute prediction semantics The prediction for inter and interintra blocks is triggered within compute_prediction. However, intra prediction is done at the transform block granularity so predict_intra is also called from transform_block. **predW** and **predH** are variables containing the smallest size that can be used for inter prediction. (This size may be increased for chroma blocks if not all blocks use inter prediction.) **predict_inter** is a function call that indicates the conceptual point where inter prediction happens. When this function is called, the inter prediction process specified in section 7.11.3 is invoked. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 188

```
compute_prediction() {
    sbMask = use_128x128_superblock ? 31 : 15
    subBlockMiRow = MiRow & sbMask
    subBlockMiCol = MiCol & sbMask
    for ( plane = 0; plane < 1 + HasChroma * 2; plane++ ) {
        planeSz = get_plane_residual_size( MiSize, plane )
        num4x4W = Num_4x4_Blocks_Wide[ planeSz ]
        num4x4H = Num_4x4_Blocks_High[ planeSz ]
        log2W = MI_SIZE_LOG2 + Mi_Width_Log2[ planeSz ]
        log2H = MI_SIZE_LOG2 + Mi_Height_Log2[ planeSz ]
        subX = (plane > 0) ? subsampling_x : 0
        subY = (plane > 0) ? subsampling_y : 0
        baseX = (MiCol >> subX) * MI_SIZE
        baseY = (MiRow >> subY) * MI_SIZE
        candRow = (MiRow >> subY) << subY
        candCol = (MiCol >> subX) << subX

        IsInterIntra = ( is_inter && RefFrame[ 1 ] == INTRA_FRAME )
        if ( IsInterIntra ) {
            if ( interintra_mode == II_DC_PRED ) mode = DC_PRED
            else if ( interintra_mode == II_V_PRED ) mode = V_PRED
            else if ( interintra_mode == II_H_PRED ) mode = H_PRED
            else mode = SMOOTH_PRED
            predict_intra( plane, baseX, baseY,
                           plane == 0 ? AvailL : AvailLChroma,
                           plane == 0 ? AvailU : AvailUChroma,
                           BlockDecoded[ plane ]
                                        [ ( subBlockMiRow >> subY ) - 1 ]
                                        [ ( subBlockMiCol >> subX ) + num4x4W ],
                           BlockDecoded[ plane ]
```



```
                                    [ ( subBlockMiRow >> subY ) + num4x4H ]
                                    [ ( subBlockMiCol >> subX ) - 1 ],
                            mode,
                            log2W, log2H )
        }
        if ( is_inter ) {
            predW = Block_Width[ MiSize ] >> subX
            predH = Block_Height[ MiSize ] >> subY
            someUseIntra = 0
            for ( r = 0; r < (num4x4H << subY); r++ )
                for ( c = 0; c < (num4x4W << subX); c++ )
                    if ( RefFrames[ candRow + r ][ candCol + c ][ 0 ] == INTRA_FRAME )
                        someUseIntra = 1
            if ( someUseIntra ) {
                predW = num4x4W * 4
                predH = num4x4H * 4
                candRow = MiRow
                candCol = MiCol
            }
            r = 0
            for ( y = 0; y < num4x4H * 4; y += predH ) {
                c = 0
                for ( x = 0; x < num4x4W * 4; x += predW ) {
                    predict_inter( plane, baseX + x, baseY + y,
                                   predW, predH,
                                   candRow + r, candCol + c)
                    c++
                }
                r++
            }
        }
    }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 83-84

7.11.3. Inter prediction process

7.11.3.1. General

The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are:

- a variable plane specifying which plane is being predicted,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted,
- variables w and h specifying the width and height of the region to be predicted,
- variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used.

The outputs of this process are predicted samples in the current frame CurrFrame.

This process is triggered by a function call to predict_inter.

The variable isCompound is set equal to RefFrames[candRow][candCol][1] > INTRA_FRAME.

The prediction arrays are formed by the following ordered steps:

1. The rounding variables derivation process specified in section 7.11.3.2 is invoked with the variable isCompound as input.
2. If plane is equal to 0 and motion_mode is equal to LOCALWARP, the warp estimation process in section 7.11.3.8 is invoked.
3. If plane is equal to 0 and motion_mode is equal to LOCALWARP and LocalValid is equal to 1, the setup shear process specified in section 7.11.3.6 is invoked with LocalWarpParams as input, and the output warpValid is assigned to LocalValid (the other outputs are discarded).
4. The variable refList is set equal to 0.
5. The variable refFrame is set equal to RefFrames[candRow][candCol][refList].

6. If (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV) and GmType[refFrame] > TRANSLATION, the setup shear process specified in section 7.11.3.6 is invoked with gm_params[refFrame] as input, and the output warpValid is assigned to globalValid (the other outputs are discarded).

7. The variable useWarp (a value of 1 indicates local warping, 2 indicates global warping) is derived as follows:

    - If w < 8 or h < 8, useWarp is set equal to 0.
    - Otherwise, if force_integer_mv is equal to 1, useWarp is set equal to 0.
    - Otherwise, if motion_mode is equal to LOCALWARP and LocalValid is equal to 1, useWarp is set equal to 1.
    - Otherwise, if all of the following are true, useWarp is set equal to 2.
        - (YMode == GLOBALMV || YMode == GLOBAL_GLOBALMV).
        - GmType[refFrame] > TRANSLATION.
        - is_scaled(refFrame) is equal to 0.
        - globalValid is equal to 1.
    - Otherwise, useWarp is set equal to 0.

8. The motion vector array mv is set equal to Mvs[candRow][candCol][refList].

9. The variable refIdx specifying which reference frame is being used is set as follows:

    - If use_intrabc is equal to 0, refIdx is set equal to ref_frame_idx[refFrame - LAST_FRAME].
    - Otherwise (use_intrabc is equal to 1), refIdx is set equal to -1 and RefFrameWidth[-1] is set equal to FrameWidth, RefFrameHeight[-1] is set equal to FrameHeight, and RefUpscaledWidth[-1] is set equal to UpscaledWidth. (These values ensure that the motion vector scaling has no effect.)

10. The motion vector scaling process in section 7.11.3.3 is invoked with plane, refIdx, x, y, mv as inputs and the output being the initial location startX, startY, and the step sizes stepX, stepY.

11. If use_intrabc is equal to 1, RefFrameWidth[-1] is set equal to MiCols * MI_SIZE, RefFrameHeight[-1] is set equal to MiRows * MI_SIZE, and RefUpscaledWidth[-1] is set equal to MiCols * MI_SIZE. (These values are needed to avoid intrabc prediction being cropped to the frame boundaries.)

12. If useWarp is not equal to 0, the block warp process in section 7.11.3.5 is invoked with useWarp, plane, refList, x, y, i8, j8, w, h as inputs and the output is merged into the 2D array preds[refList] for i8 = 0..((h-1) >> 3) and for j8 = 0..((w-1) >> 3). (Each invocation fills in a block of output of size w by h at x offset j8 * 8 and y offset i8 * 8.)

13. If useWarp is equal to 0, the block inter prediction process in section 7.11.3.4 is invoked with plane, refIdx, startX, startY, stepX, stepY, w, h, candRow, candCol as inputs and the output is assigned to the 2D array preds[refList].

14. If isCompound is equal to 1, then the variable refList is set equal to 1 and steps 5 to 13 are repeated to form the prediction for the second reference.

An array named Mask is prepared as follows:

- If compound_type is equal to COMPOUND_WEDGE and plane is equal to 0, the wedge mask process in section 7.11.3.11 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_INTRA, the intra mode variant mask process in section 7.11.3.13 is invoked with w, h as inputs.
- Otherwise if compound_type is equal to COMPOUND_DIFFWTD and plane is equal to 0, the difference weight mask process in section 7.11.3.12 is invoked with preds, w, h as inputs.
- Otherwise, no mask array is needed.

If compound_type is equal to COMPOUND_DISTANCE, the distance weights process in section 7.11.3.15 is invoked with candRow and candCol as inputs.

The inter predicted samples are then derived as follows:

- If isCompound is equal to 0 and IsInterIntra is equal to 0, CurrFrame[plane][y + i][x + j] is set equal to Clip1(preds[0][i][j]) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_AVERAGE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(preds[0][i][j] + preds[1][i][j], 1 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise if compound_type is equal to COMPOUND_DISTANCE, CurrFrame[plane][y + i][x + j] is set equal to Clip1(Round2(FwdWeight * preds[0][i][j] + BckWeight * preds[1][i][j], 4 + InterPostRound)) for i = 0..h-1 and j = 0..w-1.
- Otherwise, the mask blend process in section 7.11.3.14 is invoked with preds, plane, x, y, w, h as inputs.

If motion_mode is equal to OBMC, the overlapped motion compensation in section 7.11.3.9 is invoked with plane, w, h as inputs.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 257-259

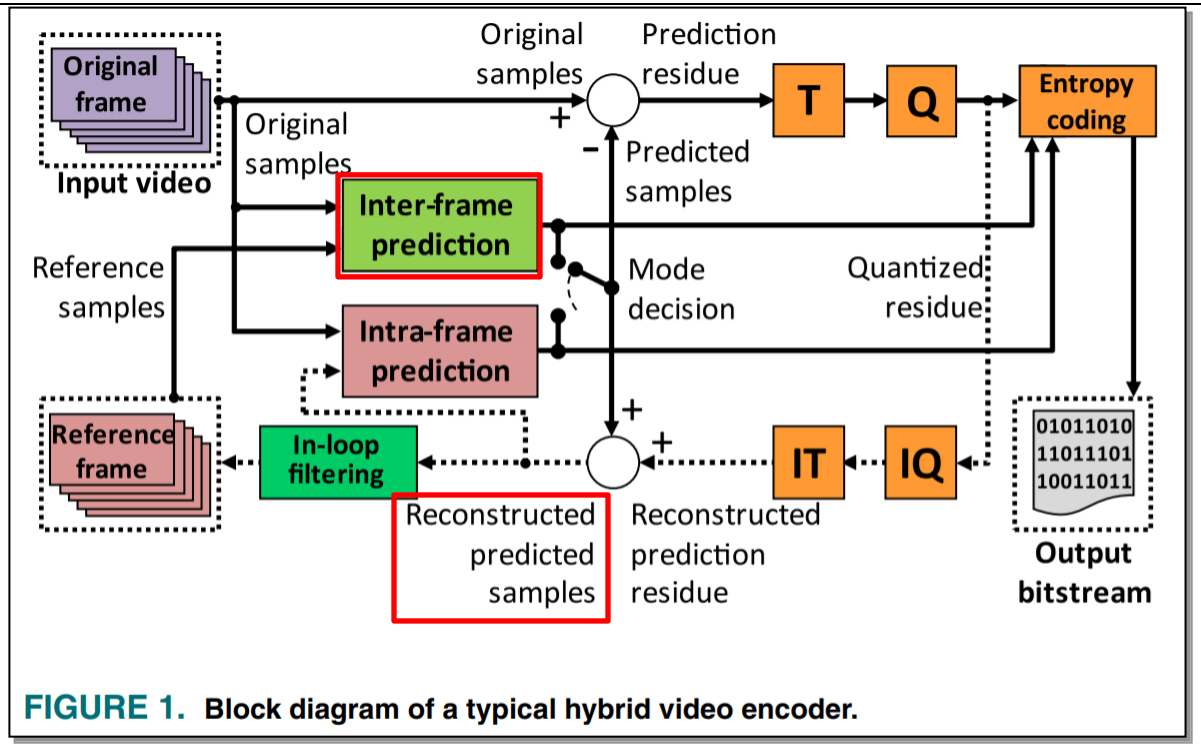
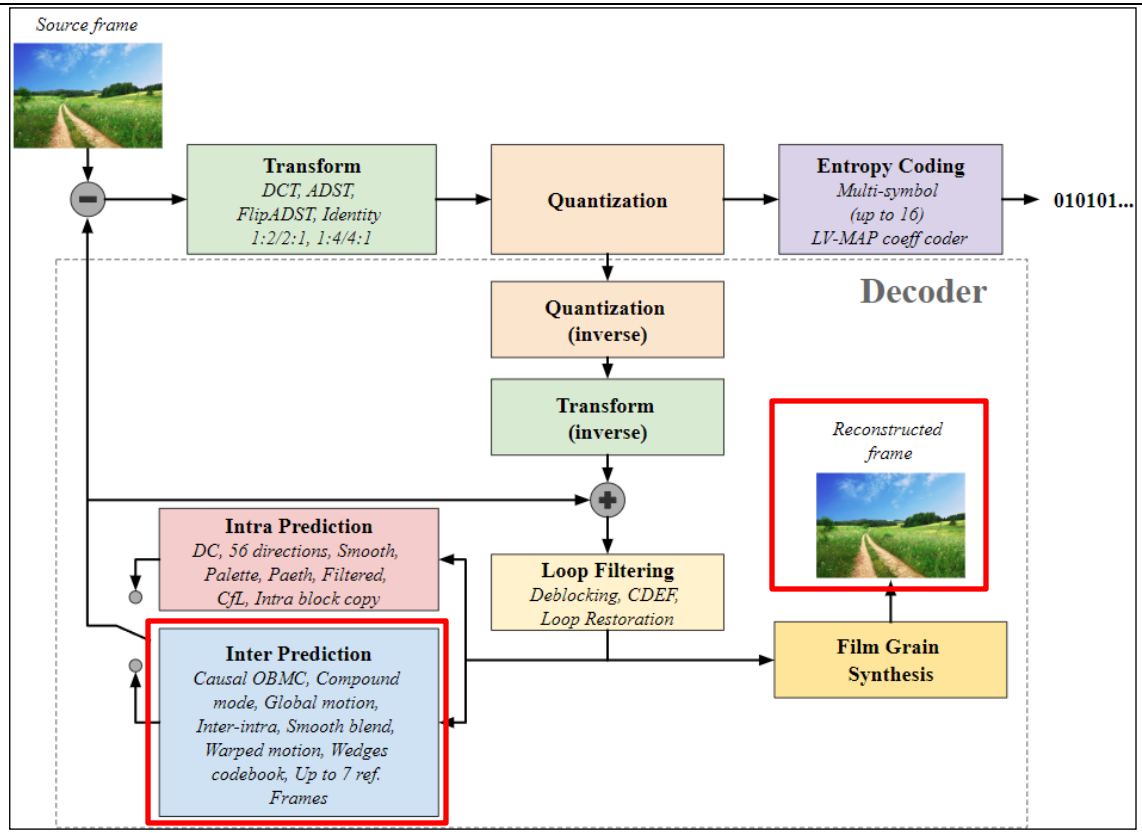


FIGURE 1. **Block diagram of a typical hybrid video encoder.**

https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf?tp=&arnumber=9536216&isnumber=9314963&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

	## 7.12.3. Reconstruct process The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35. The inputs to this process are: • a variable plane specifying which plane is being reconstructed, • variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block, • a variable txSz, specifying the size of the transform block. The outputs of this process are reconstructed samples in the current frame CurrFrame. https://aomediacodec.github.io/av1-spec/av1-spec.pdf at 294 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Based on information presently available, Advanced Coding Technologies LLC ("ACT") contends that Defendants Samsung Electronics Co. Ltd. and Samsung Electronics America, Inc. (collectively, "Defendants" or "Samsung") infringe claims 2-4 and 11 (the "Asserted Claims") of U.S. Patent No. 10,218,995 (the "'995 Patent") through the Accused Products which are manufactured, sold, offered for sale, and/or used by Samsung.

Representativeness: The Samsung 8K Neo QLED QN800A television, depicted in this chart, is believed to be representative of other televisions, display products, and laptops, including at least the AU7000, AU8000, AU9000, BU8000, BU8500, Q60A, Q60B, Q60C, Q700T, Q70A, Q70B, Q70C, Q800T, Q80A, Q80B, Q80C, Q900TS, Q90T, Q950TS, Q95T, QN700A, QN700B, QN700C, QN800A, QN800B, QN800C, QN85A, QN85B, QN85C, QN900A, QN900B, QN900C, QN90A, QN90B, QN91A, QN95A, QN95B, QN95C, S90C, S95B, S95C, The Frame 2021, The Frame 2022, The Serif 2022, Q95T, Q90T, Q80T, Q70T, Q60T, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000, LS01T, LS03A, LS03T, LS05T, LST7, QP75A-8K, LSP9T, HBU8000, HQ60B, Galaxy S23, Galaxy S23+, and Galaxy S23 Ultra, Galaxy Book Pro, Galaxy Book Pro 360, Galaxy Book2 Pro, Galaxy Book2 Pro 360, Galaxy Book2 Business 14', Galaxy Book Odyssey, Galaxy Book3 Ultra, Galaxy Book3 Pro, Galaxy Book3 Pro 360, and Galaxy Book3 360 models for the purposes of infringement of the '995 Patent. Based on publicly available information, ACT believes that the Samsung 8K Neo QLED QN800A television, charted herein, and other televisions/display products share substantially the same structure and functionality with respect to the components which are relevant to infringement given that, among other things, the listed televisions/display products all purport to support the AV1 codec. Should discovery yield documents or information demonstrating that Defendants sell other infringing products which are not represented by the Samsung 8K Neo QLED QN800A television, ACT reserves its right to amend its infringement contentions to include a chart mapping the claims of the '995 Patent to the structure and function of the newly-discovered products.

Samsung directly infringes each of the Asserted Claims by using, importing, testing, selling, and/or offering for sale the Accused Products in violation of 35 U.S.C. § 271(a).

Samsung indirectly infringes the Asserted Claims in violation of 35 U.S.C. § 271(b) by inducing third parties, including its users and/or customers, to directly infringe through their operation and use of the Accused Products. Samsung has knowingly and intentionally induced this direct infringement by, *inter alia*, (i) selling, importing, or otherwise providing the Accused Products to third parties with the intent that the Accused Products will be operated and used in a manner that practices the Asserted Claims; and (ii) marketing and advertising the Accused Products. Samsung's marketing and promotional materials for the Accused Products are

found, for example, on Samsung's website, and in App stores of operating systems for which the Accused Products are made available. For example, Samsung's website offers customers instructions and/or manuals for the Accused Products that instruct customers to, among other things, use the accused functionality in the Accused Products. Samsung's website also offers support to customers, including instruction to, among other things, use the Accused Products. On information and belief, Samsung knows that its actions will result in infringement of the Asserted Claims, or subjectively believes that there is a high probability that its actions will result in infringement of the Asserted Claims but has taken deliberate actions to avoid learning these facts.

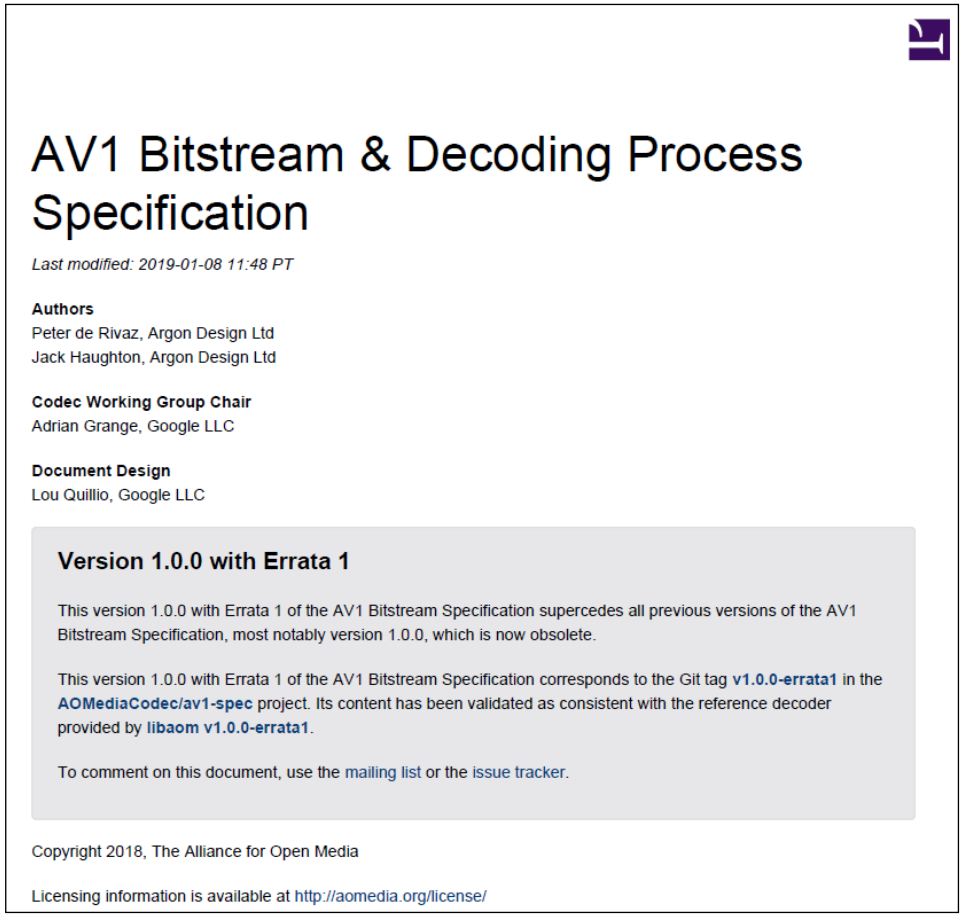
Plaintiff contends that the Accused Products perform each step of the claimed methods. Plaintiff contends that use, testing, and qualification of the Accused Products by Samsung itself, as well as use by customers and end-users of the Accused Products, perform each step of the claimed methods. On information and belief, backend servers, under the direction and control of the Accused Products, may perform certain steps of the claimed methods. Samsung, its customers, and users of the Accused Products derive benefits from their infringement.

ACT does not concede that any claims of the '995 Patent that are not listed below are not infringed by the identified Accused Products. Moreover, the citations to certain documents and other information below are intended to be exemplary only and in no way foreclose ACT from citing or relying on additional documents, information, source code, and/or testimony at a later time. These contentions are preliminary in nature, and an analysis of Samsung's products, internal documentation, source code, and/or testimony from relevant witnesses may more fully and accurately describe the infringing features of its Accused Products. Accordingly, ACT reserves the right to supplement, correct, modify, and/or amend these contentions once such additional information is made available to ACT. Furthermore, ACT reserves the right to supplement, correct, modify, and/or amend these contentions as discovery in this case progresses; in view of the Court's claim construction order(s); in view of any positions taken by Samsung including, but not limited to, positions on claim construction, invalidity, and/or non-infringement; and in connection with the preparation and exchange of expert reports.

The contents of every below claim cell on which another claim cell depends are expressly incorporated by reference in that dependent cell, as if set forth in their entirety therein.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[2P] A moving picture decoding system comprising:	The Samsung Accused Products comprise: A moving picture decoding system. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A is a moving picture decoding system that implements moving picture decoding using the AV1 codec. **Compatible Samsung TVs** Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec. **TV Series** **Models** Neo QLED 8K QN900A, QN800A, QN700A QLED 8K Q950T, Q900T, Q800T, Q700T Neo QLED 4K QN95A, QN91A, QN90A, QN85A QLED 4K Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T Crystal UHD AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000 The Serif LS01T The Frame LS03A, LS03T https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/ The AV1 codec is used for video decoding, which is moving picture decoding.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

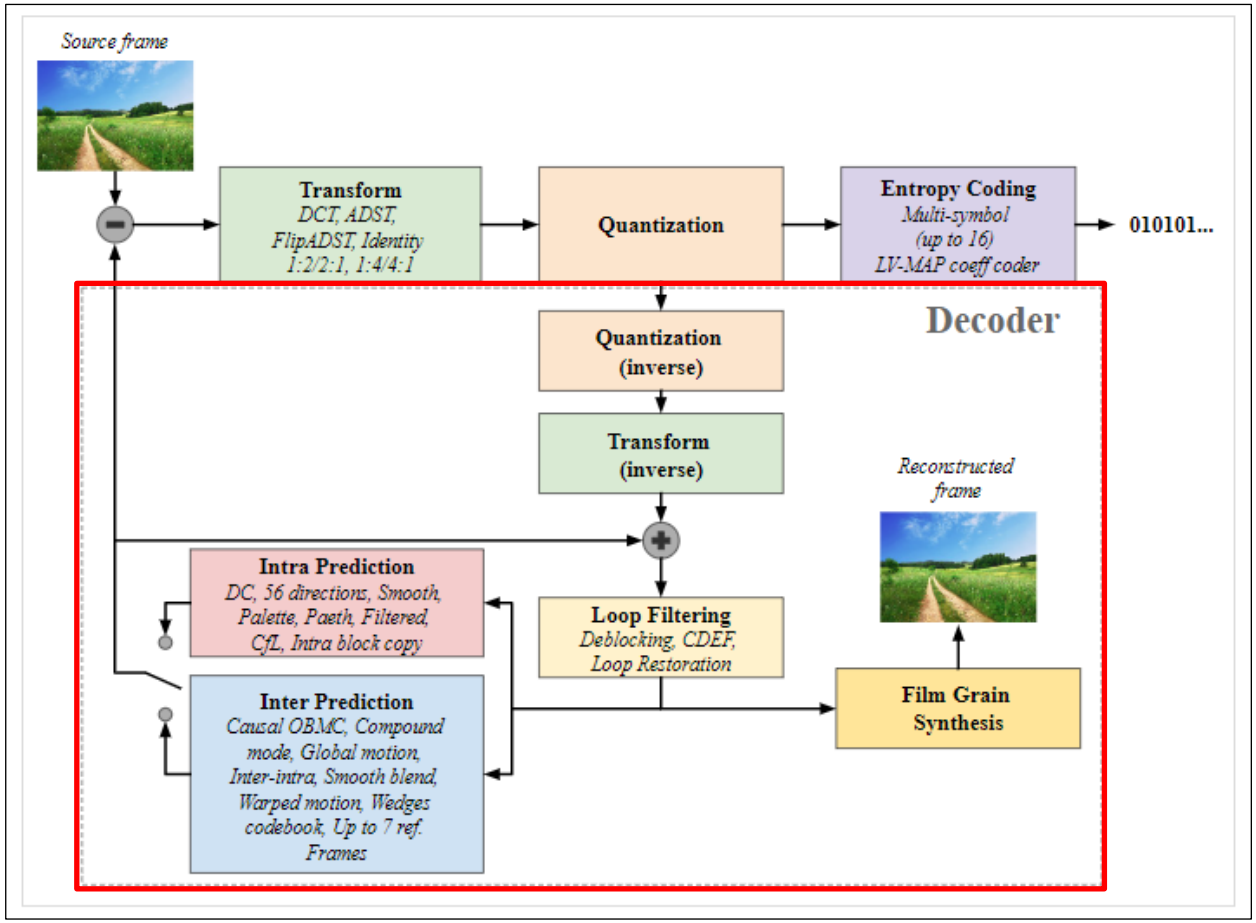
The Samsung 8K Neo QLED QN800A television supports the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Supported video codecs (QN7**A Series or higher)

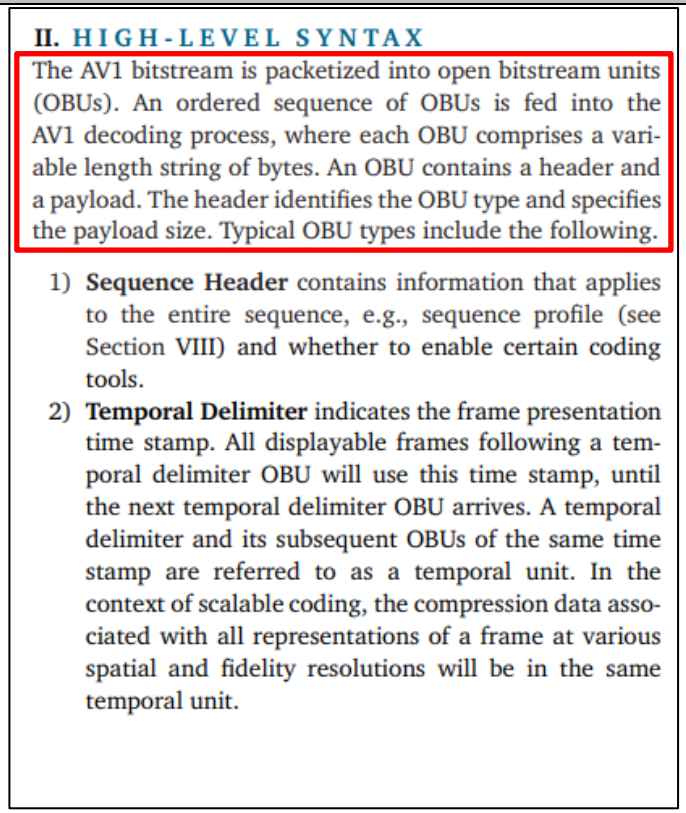
File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	The AV1 decoder reproduces reconstructed frames.  https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2A] a demultiplexer configured to work on a sequence of input encoded bits to implement a process for a prescribed demultiplexing to output at least a first and a second sequence of encoded bits;	The Samsung Accused Products comprise: A demultiplexer configured to work on a sequence of input encoded bits to implement a process for a prescribed demultiplexing to output at least a first and a second sequence of encoded bits. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A includes a demultiplexer configured to work on a sequence of input encoded bits (*e.g.*, "[t]he AV1 bitstream"). The AV1 bitstream that is input into the demultiplexer "is packetized into open bitstream units (OBUs)," which are at least a first and a second sequence of encoded bits.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.

https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Samsung's 8K Neo QLED QN800A demultiplexer is configured to implement a process for a prescribed demultiplexing to output at least a first sequence of encoded bits (*e.g.*, sequence of intraframe type bits) and a second sequence of encoded bits (*e.g.*, sequence of interframe type bits). The information regarding the type of bit (and, consequently, whether it falls within the first or second sequence of encoded bits) is located in the

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	frame headers of the various Open Bitstream Units (OBUs), "which provides identifying information for the contained data (payload)." **OBU** All structures are packetized in "Open Bitstream Units" or OBUs. Each OBU has a header, which provides identifying information for the contained data (payload). https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method. 4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering. 5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost. 6) **Metadata** carries information, such as high dynamic range, scalability, and timecode. 7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13]. We refer to [9] for bit field definitions and more detailed consideration of high-level syntax. https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

5.9. Frame header OBU syntax

5.9.2. Uncompressed header syntax

uncompressed_header( ) {	Type
if (frame_id_numbers_present_flag) {	
idLen = (additional_frame_id_length_minus_1 +	
delta_frame_id_length_minus_2 + 3)	
}	

allFrames = (1 << NUM_REF_FRAMES) - 1	
if (reduced_still_picture_header) {	
show_existing_frame = 0	
frame_type = KEY_FRAME	
FrameIsIntra = 1	
show_frame = 1	

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

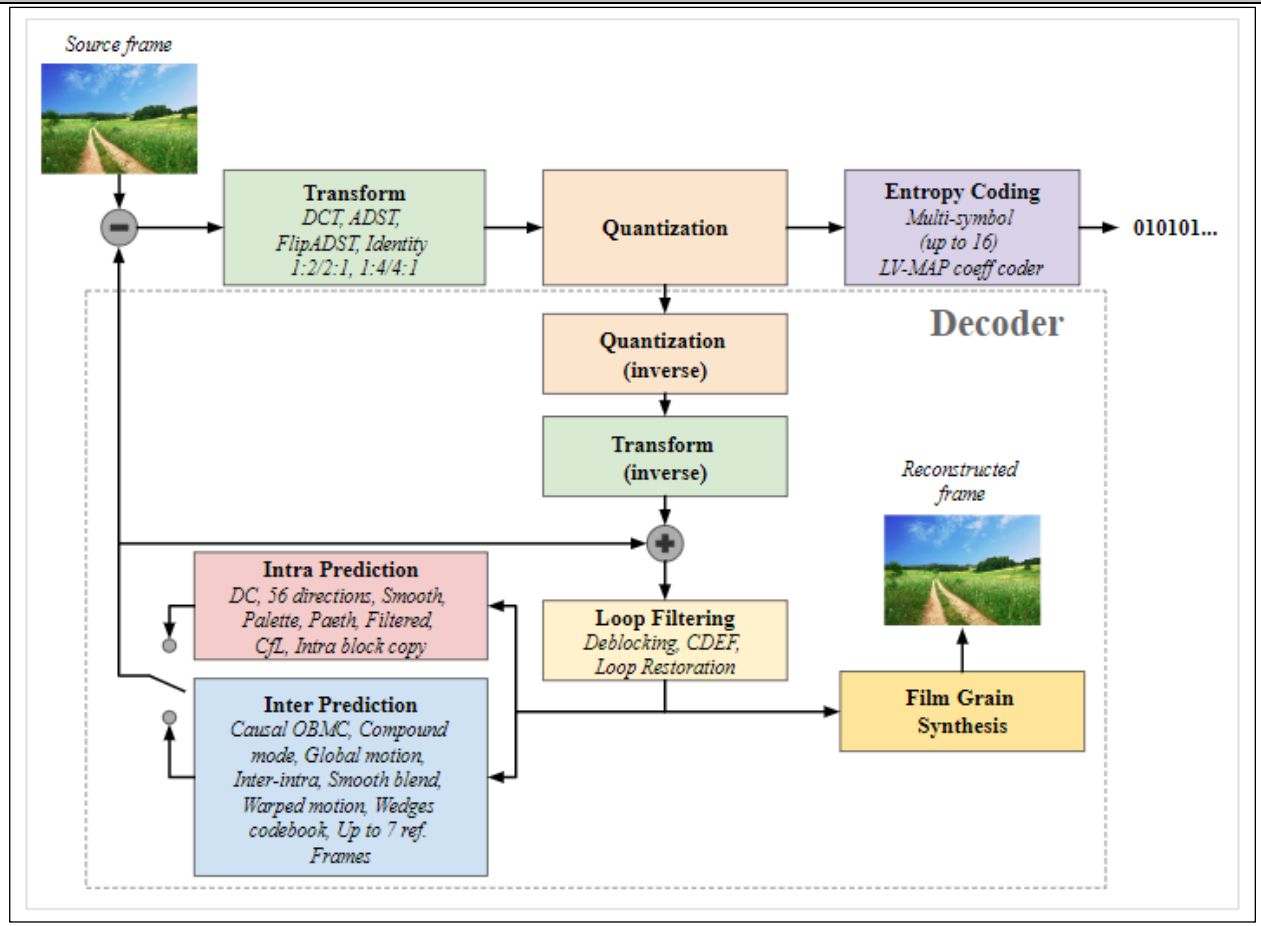
6.8.2. Uncompressed header semantics

frame_type specifies the type of the frame:

frame_type	Name of frame_type
0	KEY_FRAME
1	INTER_FRAME
2	INTRA_ONLY_FRAME
3	SWITCH_FRAME

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Inter frame A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Intra frame** A frame compressed using only intra prediction which can be independently decoded. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2B] a first decoder configured to acquire the first sequence of encoded bits obtained with a standard resolution at the demultiplexer to implement thereon a process for a prescribed first	The Samsung Accused Products comprise: A first decoder configured to acquire the first sequence of encoded bits obtained with a standard resolution at the demultiplexer to implement thereon a process for a prescribed first decoding to create a sequence of decoded pictures with a standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses the AV1 codec, which comprises a first decoder configured to acquire the first sequence of encoded bits obtained with a standard resolution (*e.g.*, sequence of intraframes) at the demultiplexer to implement thereon a process for a prescribed first decoding to create a sequence of decoded pictures with a standard resolution (*e.g.*, a sequence of decoded intraframes).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
decoding to create a sequence of decoded pictures with a standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, the sequence of decoded pictures with a standard resolution can be super-resolved or non-super-resolved depending on the use_superres bit information sent within each frame header. When use_superres is equal to 1, upscaling is needed. When use_superres is not equal to 1, the first decoder is configured to implement a process that creates a sequence of decoded pictures where upscaling is not needed (*i.e.*, a sequence of decoded pictures with a standard resolution). 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2C] a first super-resolution enlarger configured to acquire the sequence of decoded	The Samsung Accused Products comprise: A first super-resolution enlarger configured to acquire the sequence of decoded pictures created with a standard resolution at the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
pictures created with a standard resolution at the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution;	For example, the Samsung 8K Neo QLED QN800A comprises a first super-resolution enlarger configured to (for intraframe sequences for which super-resolution is active) acquire the sequence of decoded pictures created with a standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframe bits) to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement (*e.g.*, through upscaling and loop restoration) to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution. As described above with respect to Claim [2B] and replicated below, when use_superres is equal to 1, upscaling (*i.e.*, a first super-resolution) is needed for the sequence of decoded pictures (*e.g.*, pictures based on the intraframe bit sequence at standard resolution). **6.8.7. Superres params semantics** **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf As noted below, the AV1 codec utilizes a "frame super-resolution framework" that comprises two steps: (1) "Linear upscale" and (2) "Loop restore".

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

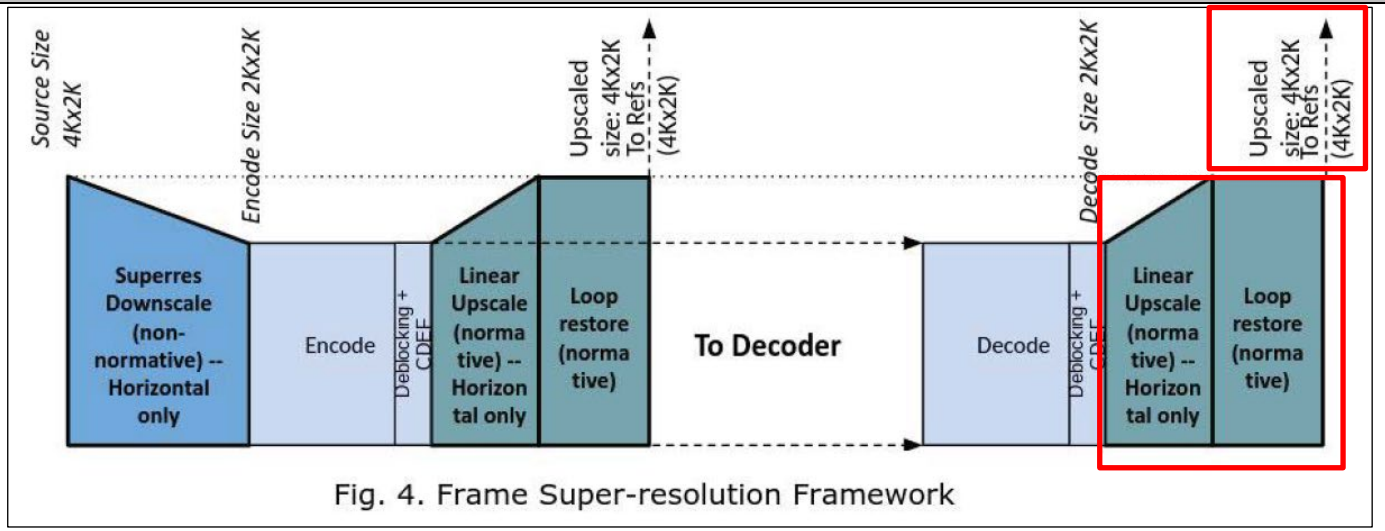


Fig. 4. Frame Super-resolution Framework

https://ieeexplore.ieee.org/document/8954553

7.16. Upscaling process

Input to this process is an array inputFrame of width FrameWidth and height FrameHeight.

The output of this process is a horizontally upscaled frame of width UpscaledWidth and height FrameHeight.

If use_superres is equal to 0, no upscaling is required and this process returns inputFrame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

This process is specified as:

```
for ( plane = 0; plane < NumPlanes; plane++ ) {
    if ( plane > 0 ) {
        subX = subsampling_x
        subY = subsampling_y
    } else {
        subX = 0
        subY = 0
    }
    downscaledPlaneW = Round2(FrameWidth, subX)
    upscaledPlaneW = Round2(UpscaledWidth, subX)
    planeH = Round2(FrameHeight, subY)
    stepX = ((downscaledPlaneW << SUPERRES_SCALE_BITS) + (upscaledPlaneW / 2)) / upscaledPlaneW
    err = (upscaledPlaneW * stepX) - (downscaledPlaneW << SUPERRES_SCALE_BITS)
    initialSubpelX =
      (-((upscaledPlaneW - downscaledPlaneW) << (SUPERRES_SCALE_BITS - 1)) + upscaledPlaneW / 2) /
upscaledPlaneW +
      (1 << (SUPERRES_EXTRA_BITS - 1)) - err / 2
    initialSubpelX &= SUPERRES_SCALE_MASK
    miW = MiCols >> subX
    minX = 0
    maxX = miW * MI_SIZE - 1
    for ( y = 0; y < planeH; y++ ) {
        for ( x = 0; x < upscaledPlaneW; x++ ) {
            srcX = -(1 << SUPERRES_SCALE_BITS) + initialSubpelX + x*stepX
            srcXPx = (srcX >> SUPERRES_SCALE_BITS)
            srcXSubpel = (srcX & SUPERRES_SCALE_MASK) >> SUPERRES_EXTRA_BITS
            sum = 0
            for ( k = 0; k < SUPERRES_FILTER_TAPS; k++ ) {
                sampleX = Clip3(minX, maxX, srcXPx + (k - SUPERRES_FILTER_OFFSET))
                px = frame[plane][y][sampleX]
                sum += px * Upscale_Filter[srcXSubpel][k]
            }
            outputFrame[plane][y][x] = Clip1(Round2(sum, FILTER_BITS))
        }
    }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17. Loop restoration process

Input to this process are the arrays UpscaledCurrFrame (of reconstructed samples) and UpscaledCdefFrame (of deringed samples).

Output from this process is the array LrFrame of loop restored samples.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

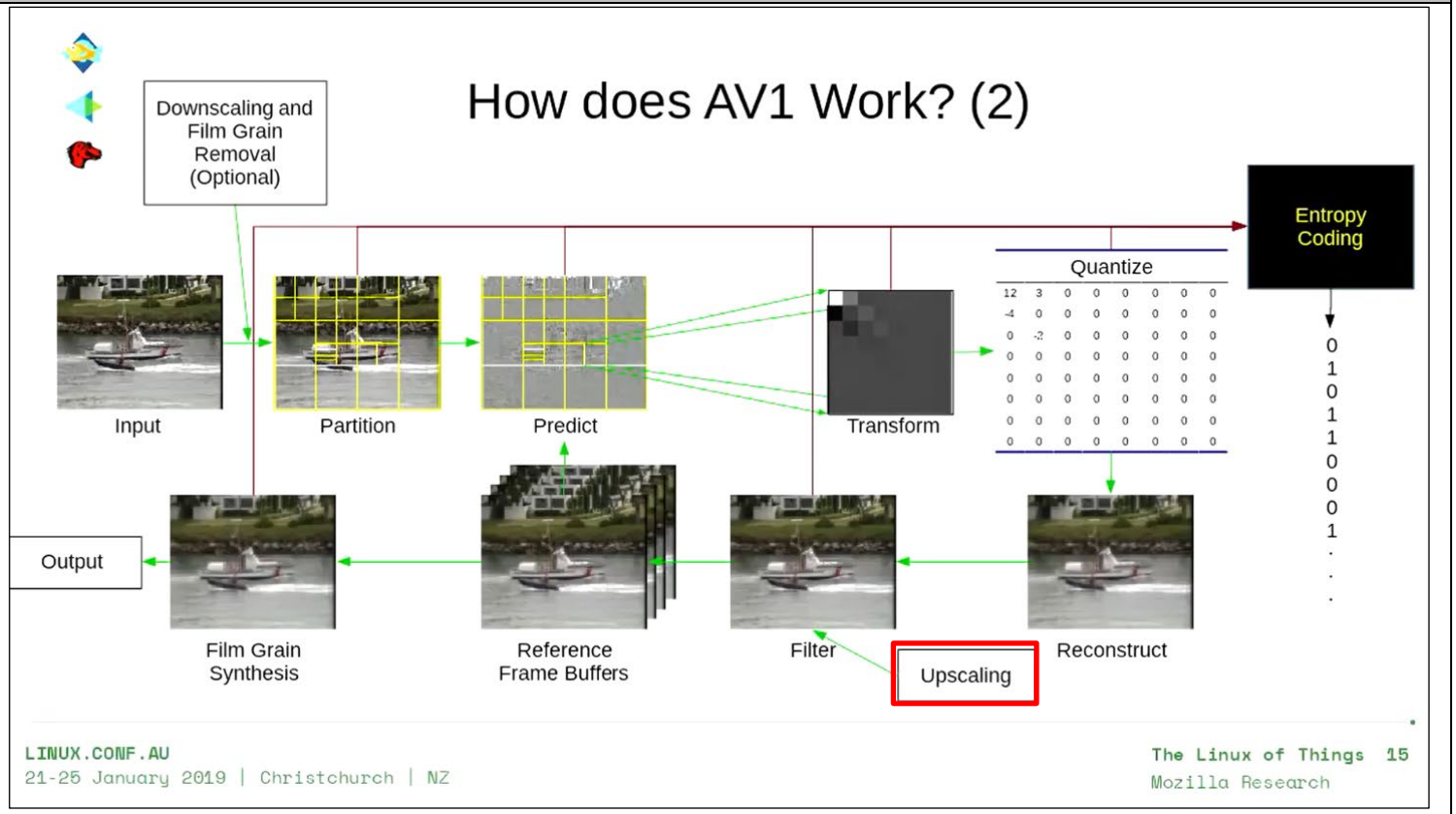
The array LrFrame is set equal to a copy of UpscaledCdefFrame. (The contents of LrFrame will later be overwritten for blocks that require restoration filtering.)

If UsesLr is equal to 0, then the process returns immediately after performing this copy.

Otherwise, loop restoration is applied as follows:

```
for ( y = 0; y < FrameHeight; y += MI_SIZE ) {
  for ( x = 0; x < UpscaledWidth; x += MI_SIZE ) {
    for ( plane = 0; plane < NumPlanes; plane++ ) {
      if ( FrameRestorationType[ plane ] != RESTORE_NONE ) {
        row = y >> MI_SIZE_LOG2
        col = x >> MI_SIZE_LOG2
        loop_restore_block( plane, row, col )
      }
    }
  }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, an interpolation of pixels is applied on the decoded intraframe pictures to create a super resolution enlarged decoded picture. In AV1, the interpolation process is performed by wiener filter process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17.1. Loop restore block process

The inputs to this process are:

- a variable plane specifying whether the process is filtering Y, U, or V samples,
- variables row and col specifying the location of the block in units of 4x4 blocks in the upscaled luma plane.

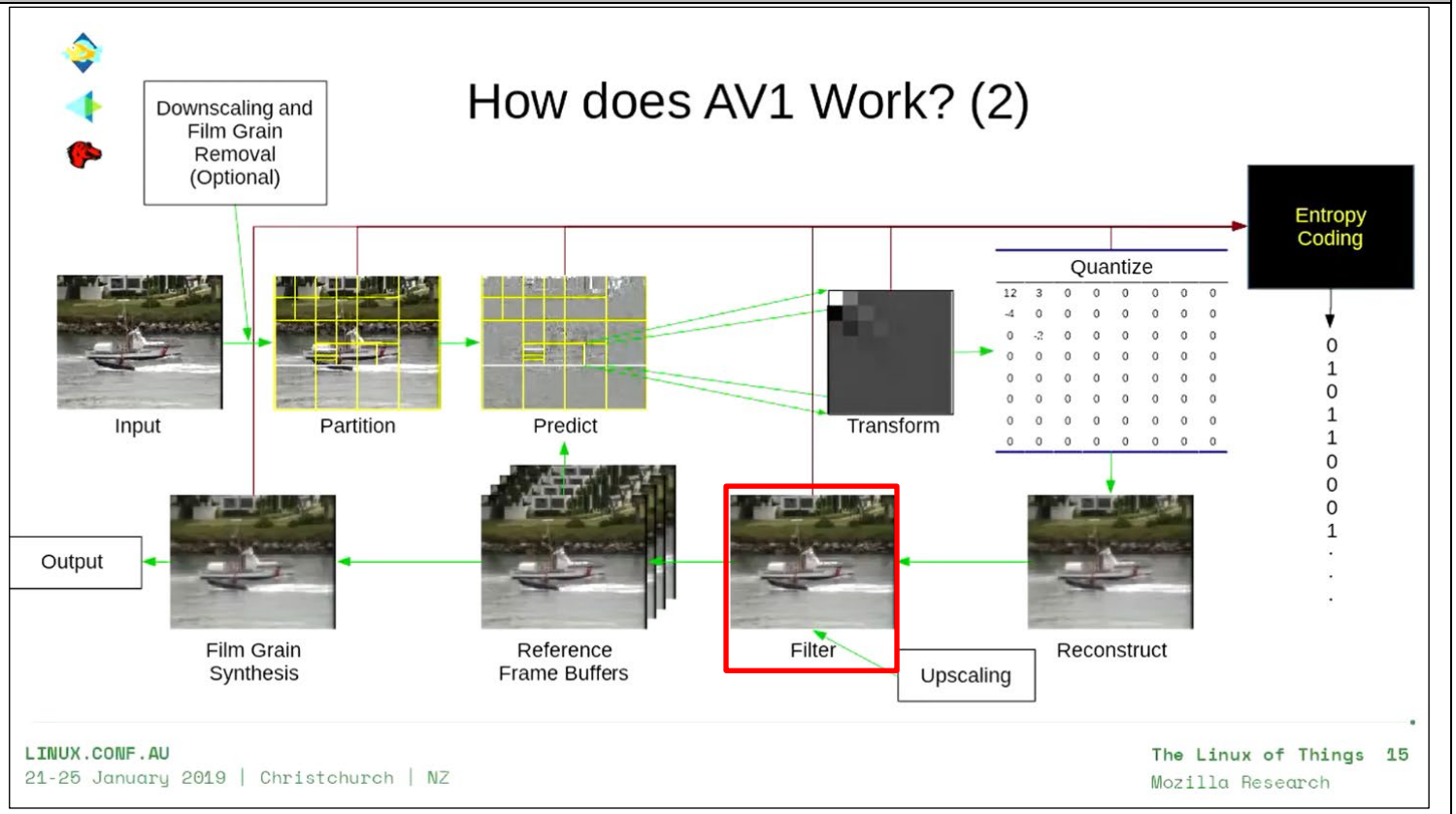
The output of this process are samples in LrFrame[plane].

The filter to used depends on rType as follows:

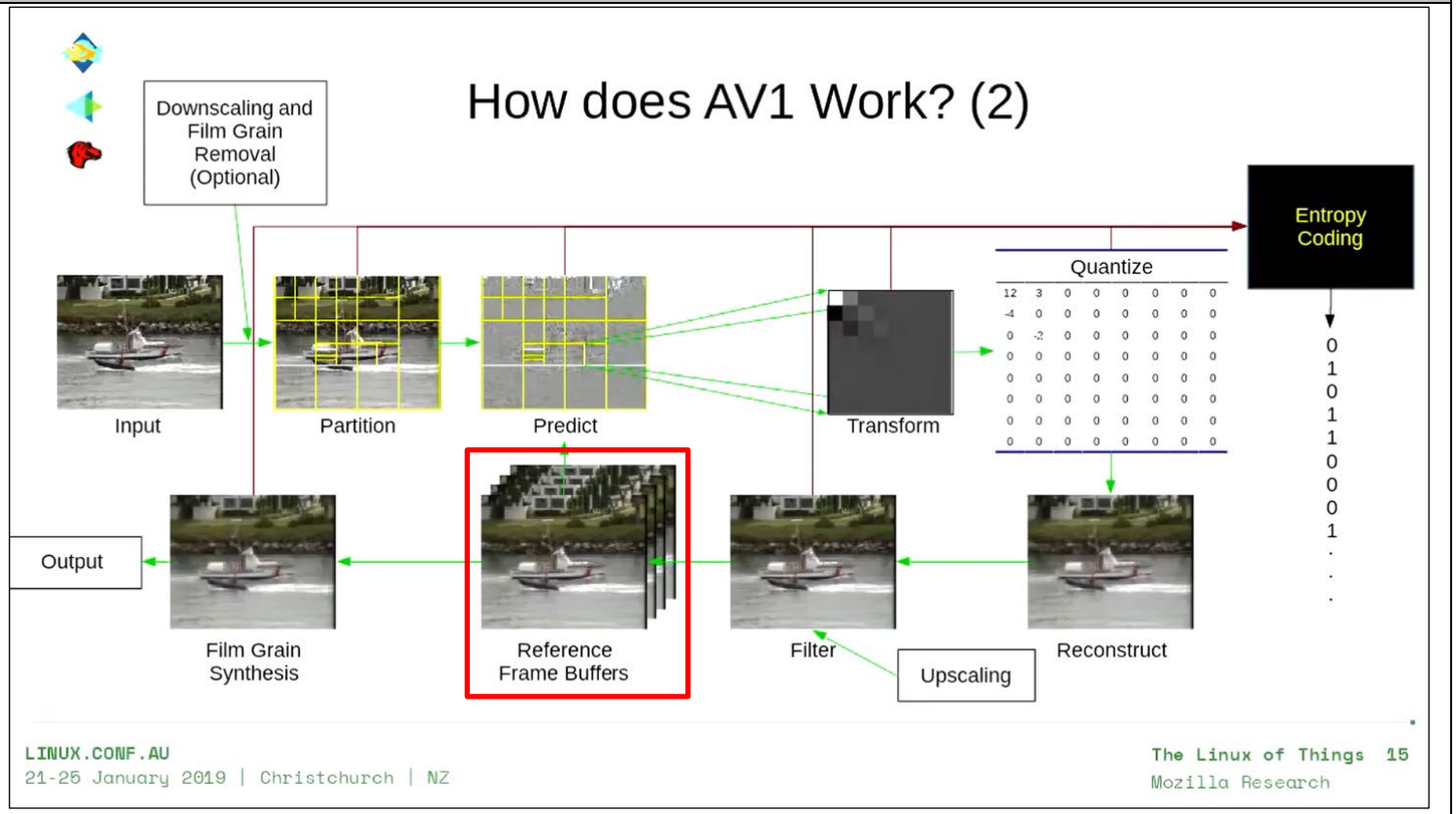
- If rType is equal to RESTORE_WIENER, the Wiener filter process specified in section 7.17.4 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise, if rType is equal to RESTORE_SGRPROJ, the self guided filter process specified in section 7.17.2 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise (rType is equal to RESTORE_NONE), no filtering is applied.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	7.17.4. Wiener filter process The inputs to this block are: • a variable plane specifying whether the process is filtering Y, U, or V samples, • variables unitRow and unitCol specifying the position of the loop restoration unit, • variables x and y specifying the position of the block in samples relative to the top-left corner of the current plane, • variables w and h specifying the size of the block in samples. The output from this process are modified samples in LrFrame. The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[2D] a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger to work on the sequence of super-resolution enlarged decoded pictures to implement a process for a prescribed resolution conversion to create a sequence of super-resolution decoded	The Samsung Accused Products comprise: a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger to work on the sequence of super-resolution enlarged decoded pictures to implement a process for a prescribed resolution conversion to create a sequence of super-resolution decoded pictures with a standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger. After the loop restoration process, the reconstructed intraframes are added to the reference buffer list which are further used for decoding interframes. The references pictures at the decoding side are scaled according to the resolution of the current interframe to be decoded. Since the first super-resolution enlarger provides an upscaled decoded reference picture, the reference pictures are downscaled to match the current interframe's resolution to be used as a reference picture for those interframes.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
pictures with a standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

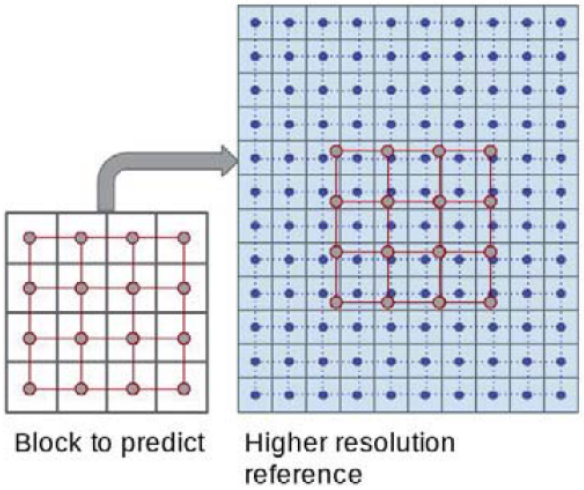
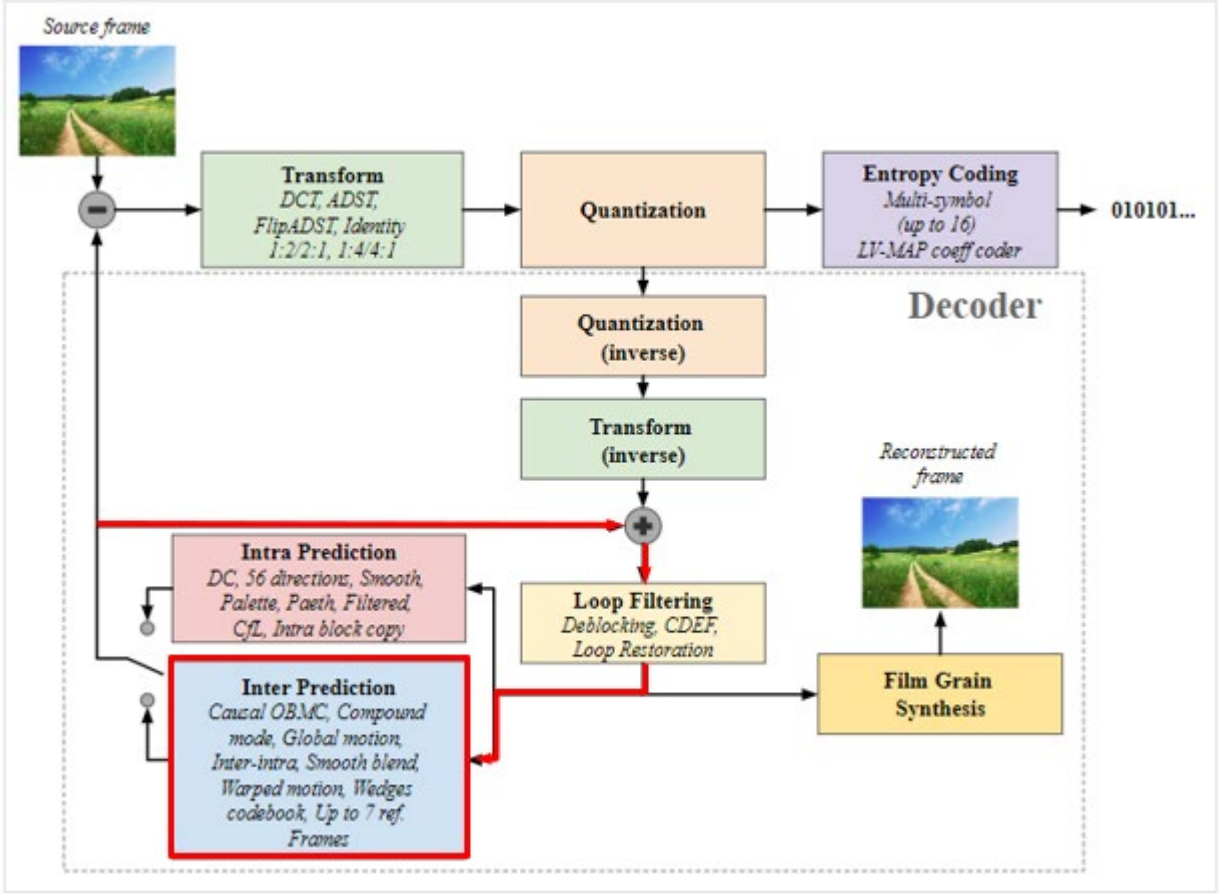


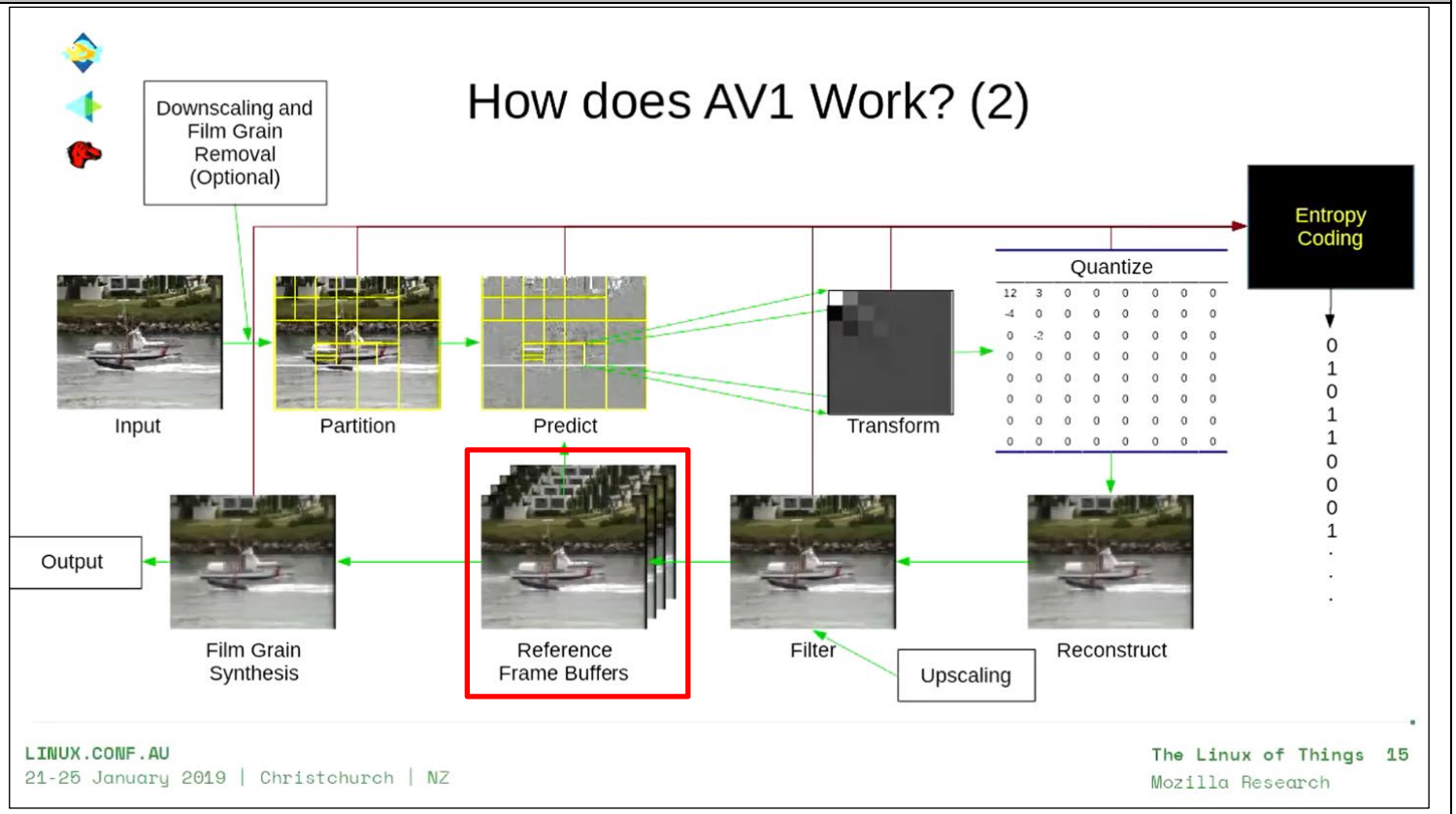
Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2E] a second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder as a set of first reference pictures,	The Samsung Accused Products comprise: A second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer (*e.g.*, sequence of interframes) as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframes) as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures (*e.g.*, intraframes that are downscaled to match the current interframe's resolution to be used as a reference picture). As described above with respect to Claim Element [2A], the demultiplexer outputs two sequences of encoded bits, where the second sequence of encoded bits is the sequence of interframe bits. This sequence of interframe bits is a set of decoding targets because reconstructed interframes without super-resolution active (i.e., the second sequence of encoded bits obtained with a standard resolution) are used for inter-prediction of the current frame in the decoding process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures, and	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, as described above with respect to Claim Element [2B], the first decoder creates a sequence of decoded pictures (based on the first sequence of intraframe bits) created with the standard resolution at the first decoder

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	as a set of first reference pictures since these intraframes create the reference frames for the inter prediction described above. 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM.[7] https://aomediacodec.github.io/av1-spec/av1-spec.pdf Moreover, as described above with respect to Claim Elements [2C] and [2D], a first resolution converter converts the sequence of super-resolution decoded pictures with the standard resolution. The AV1 decoder pipeline, with super-resolution active, produces decoded frames which are references that are super-resolved and then downscaled to match the current frame resolution (*i.e.*, the set of super-resolution enlarged and converted decoded pictures from the second resolution converter that acts as a set of second reference pictures).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

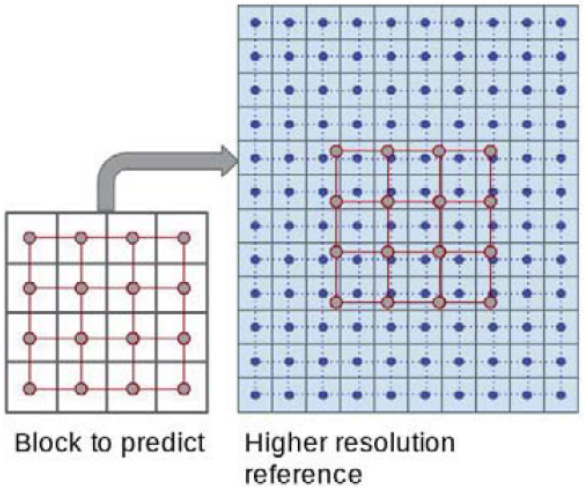
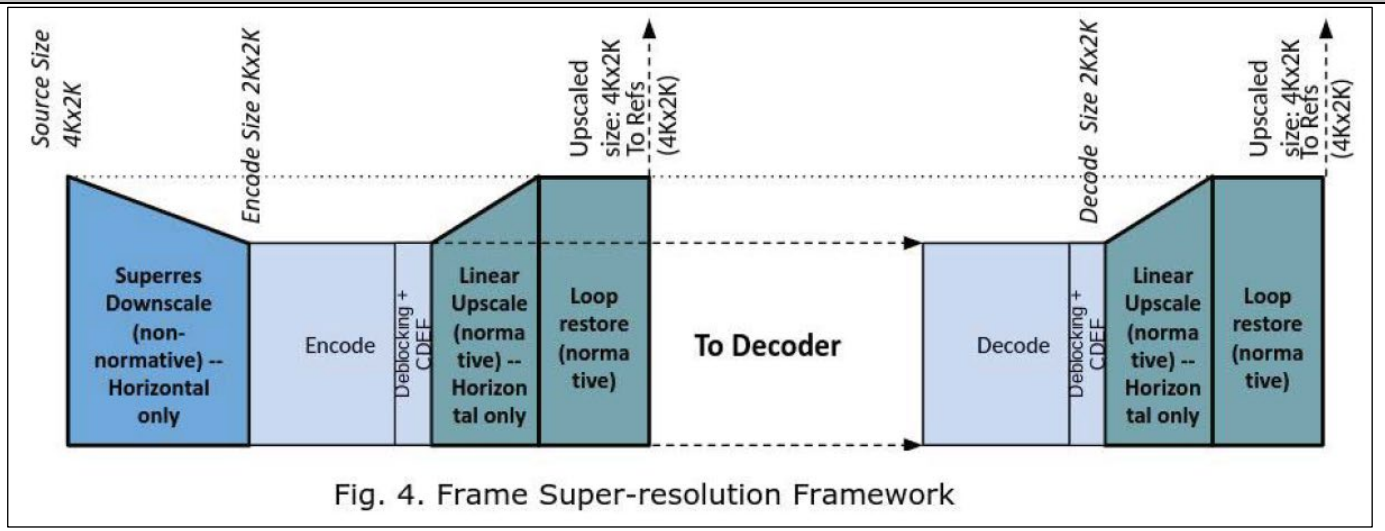


Fig. 2. Predicting a block from different (higher shown) resolution reference

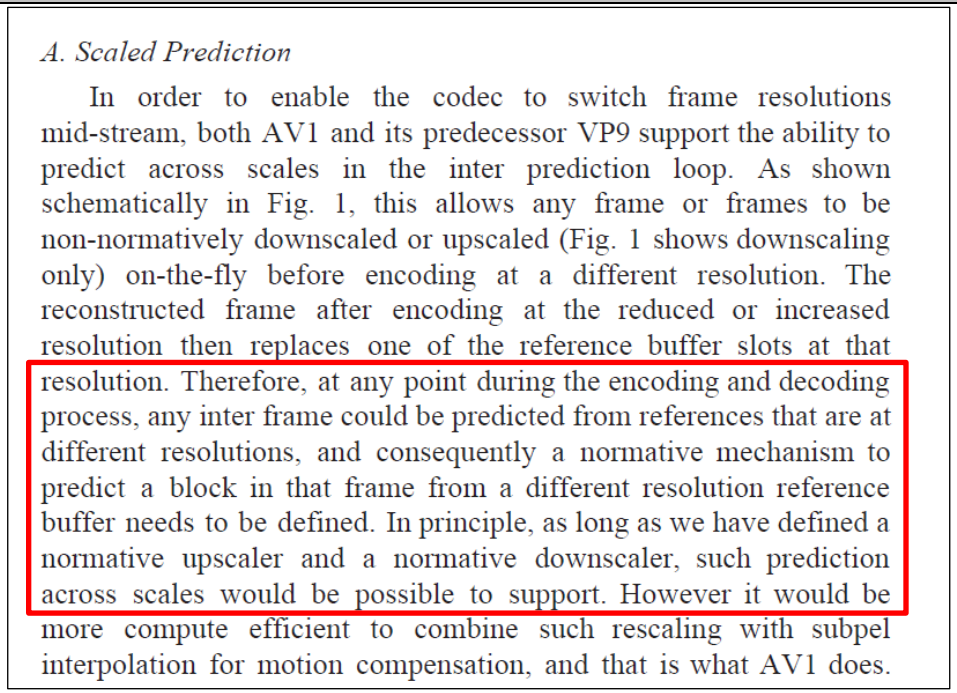
https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Therefore, both the sequence of decoded pictures created initially at standard resolution and the sequence of decoded pictures created by first upscaling and then downscaling, are sets of reference pictures acquired by the second decoder along with the second sequence comprising interframe pictures to perform the requisite interframe prediction. *B.1 Framework At a Glance* Fig. 4 depicts the overall in-loop filtering pipeline in AV1 and the super-resolution framework. According to this framework, on the encoder side, a source frame may be first downscaled non-normatively and encoded at a lower resolution. The deblocking filter, and another filter called Constrained Directional Enhancement Filter (CDEF) [17] supported in AV1, are then used to remove blocking and ringing artifacts while preserving edges, at the lower resolution. This is followed by a normative Linear Upsampling

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	process to full source resolution. Finally, another AV1 tool called Loop Restoration (LR) [18] is applied at full resolution to recover the high frequencies lost during the prior downscaling and quantization processes. The loop restored frame updates the reference buffer store for encoding subsequent frames. A similar process is followed on the decoder side as well. Note that the Deblocking, CDEF, Linear Upscaling and LR in order are all part of the normative in-loop filtering process in AV1, where each filter can be turned on or off per frame. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Fig. 4. Frame Super-resolution Framework https://ieeexplore.ieee.org/document/8954553 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2F] a second decoder configured to…select one of the set of first reference pictures and the set of second	The Samsung Accused Products comprise: A second decoder configured to select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution based on the set of decoding targets and the set of selected reference pictures. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution based on the set of decoding targets and the set of selected	For example, the Samsung 8K Neo QLED QN800A comprises a second decoder configured to select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information (*e.g.*, selected based on a reference index) to implement a combination of processes for a prescribed prediction (*e.g.*, reconstructed frames without super-resolution are used for inter prediction of the current frame) and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution (*e.g.*, downscaled) based on the set of decoding targets (*e.g.*, resolution of the current frame) and the set of selected reference pictures (*e.g.*, the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index). As shown below, the AV1 codec supports the ability to predict pictures across scales using upscalers and downscalers (*i.e.*, a reference picture at a first resolution can be used to predict a picture at a second resolution). The second decoder waits for the current interframe to be decoded as received from the demultiplexer. When it receives the Interframe, the frame can be decoded based on the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
reference pictures; and	 *A. Scaled Prediction* In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553 Since AV1 specification allow each frame to either be normally decoded or decoded with super-resolve steps, it is implied that the reference picture buffer set consists of both non-super-resolved and super-resolved intraframes that can be used as reference pictures. For the second decoder to decode the current interframe, the reference frame is selected based on the reference index. Thus, the reference index that indicates which reconstructed frame is selected for prediction is implied as 'the reference picture selection information' that is sent in the encoded bitstream.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	ref_frame_idx[i] specifies which reference frames are used by inter frames. It is a requirement of bitstream conformance that RefValid[ref_frame_idx[i]] is equal to 1, and that the selected reference frames match the current frame in bit depth, profile, chroma subsampling, and color space. **Note:** Syntax elements indicate a reference (such as LAST_FRAME, ALTREF_FRAME). These references are looked up in the ref_frame_idx array to find which reference frame should be used during inter prediction. There is no requirement that the values in ref_frame_idx should be distinct. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2G] a second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder to work on the sequence of	The Samsung Accused Products comprise: A second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder (*e.g.*, a sequence of decoded intraframes) to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution (*e.g.*, a sequence of upscaled decoded intraframes).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution,	Different 8-tap filter coefficients can be used for upscaling of the decoded frame. Google Git aomedia / aom / refs/heads/main / . / av1 / common / **resize.c** blob: f4bfcd062af2a6c4643314ec1501aa9c27306aa2 [file] [log] [blame] 1 /* 2 * Copyright (c) 2016, Alliance for Open Media. All rights reserved 3 * 4 * This source code is subject to the terms of the BSD 2 Clause License and 5 * the Alliance for Open Media Patent License 1.0. If the BSD 2 Clause License 6 * was not distributed with this source code in the LICENSE file, you can 7 * obtain it at www.aomedia.org/license/software. If the Alliance for Open 8 * Media Patent License 1.0 was not distributed with this source code in the 9 * PATENTS file, you can obtain it at www.aomedia.org/license/patent. 10 */

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

```
176  const int16_t av1_resize_filter_normative[(
177      1 << RS_SUBPEL_BITS)][UPSCALE_NORMATIVE_TAPS] = {
178  #if UPSCALE_NORMATIVE_TAPS == 8
179    { 0, 0, 0, 128, 0, 0, 0, 0 },        { 0, 0, -1, 128, 2, -1, 0, 0 },
180    { 0, 1, -3, 127, 4, -2, 1, 0 },      { 0, 1, -4, 127, 6, -3, 1, 0 },
181    { 0, 2, -6, 126, 8, -3, 1, 0 },      { 0, 2, -7, 125, 11, -4, 1, 0 },
182    { -1, 2, -8, 125, 13, -5, 2, 0 },    { -1, 3, -9, 124, 15, -6, 2, 0 },
```

https://aomedia.googlesource.com/aom/+/refs/heads/main/av1/common/resize.c

The AV1 codec accomplishes prediction across frames of different resolutions through its normative upscaling filters.

For example, intraframes or Alternate Reference Frames, which are normally decoded by the first decoder, may require upscaling to match resolution of current frame. Since the AV1 codec allows switching between resolutions and using references of different resolutions, and coding can be done in an arbitrary resolution, a reconstructed frame which is to be used as reference might be present in a reconstructed-reference buffer whose resolution is different (lower) than the resolution of the interframe currently being decoded, for which the intraframe is an indexed reference. In such a scenario, a resolution conversion (*e.g.*, an upscale) would be required for decoding the current interframe. Such a resolution upscale is the AV1 codec's second resolution converter.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

https://ieeexplore.ieee.org/document/8954553

Note that the AV1 bit-stream is flexible enough that each coded frame can use (a) horizontal-only super-resolution mode as described above, (b) code a frame at an arbitrary resolution within the ratio constraints, (c) a combination of both, or (d) no scaling at all. An intelligent encoder can utilize these options as additional knobs to be

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[2H] wherein the set of decoding targets, the set of first reference pictures, and the set of second reference pictures have the same value in spatial resolution.	The set of decoding target pictures, the set of first reference pictures, and the set of second reference pictures in the Samsung Accused Products have the same value in spatial resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A upscales or downscales the reference pictures to match the resolution of the decoding targets. That is, the AV1 codec in the Samsung 8K Neo QLED QN800A scales (1) the set of decoding target pictures; (2) the set of first reference pictures, and (3) the set of second reference pictures to have the same value in spatial resolution. As described below, "the motion vectors transmitted in the bitstream are always at the source resolution."

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

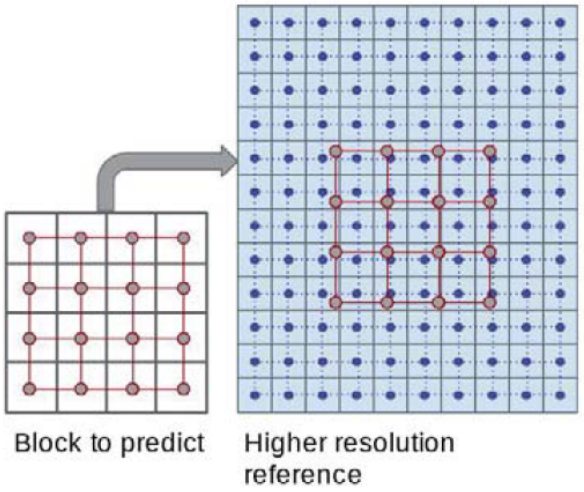


Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3P] A moving picture decoding method comprising:	Users of the Samsung Accused Products perform a moving picture decoding method. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A is a moving picture decoding system that implements moving picture decoding using the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Compatible Samsung TVs

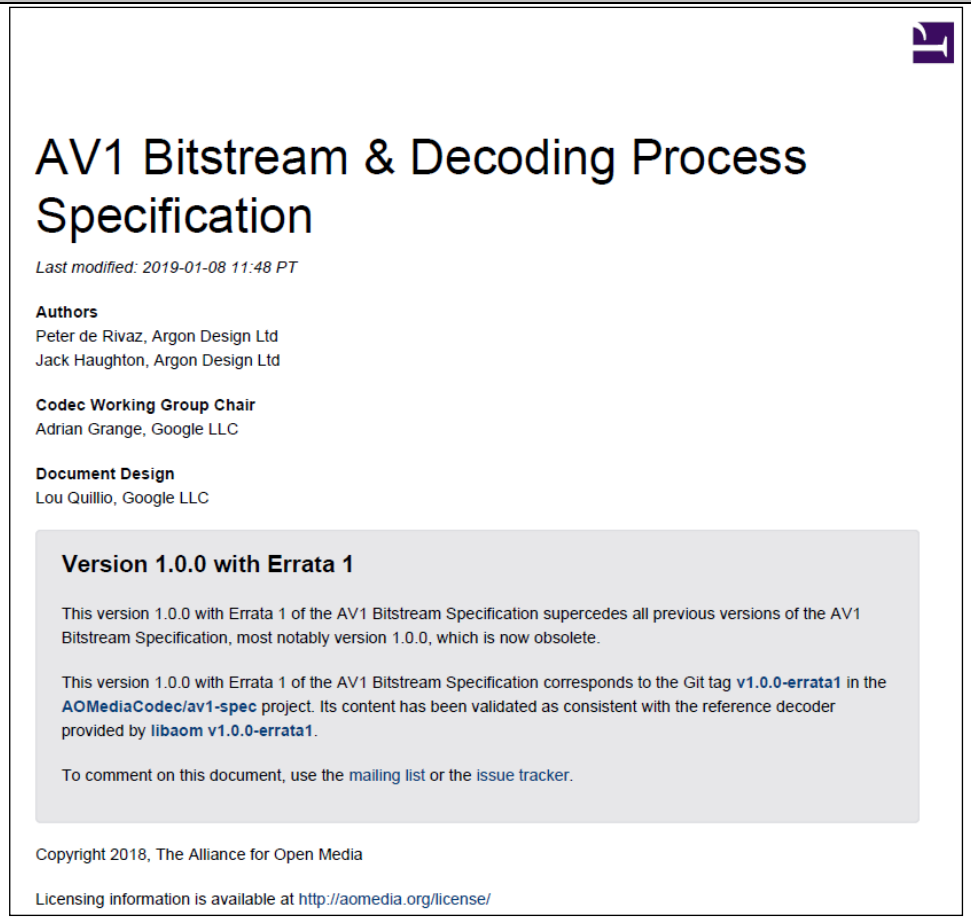
Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec.

TV Series	Models
Neo QLED 8K	QN900A, QN800A, QN700A
QLED 8K	Q950T, Q900T, Q800T, Q700T
Neo QLED 4K	QN95A, QN91A, QN90A, QN85A
QLED 4K	Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T
Crystal UHD	AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000
The Serif	LS01T
The Frame	LS03A, LS03T

https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

The AV1 codec is used for video decoding, which is moving picture decoding.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

The Samsung 8K Neo QLED QN800A television supports the AV1 codec.

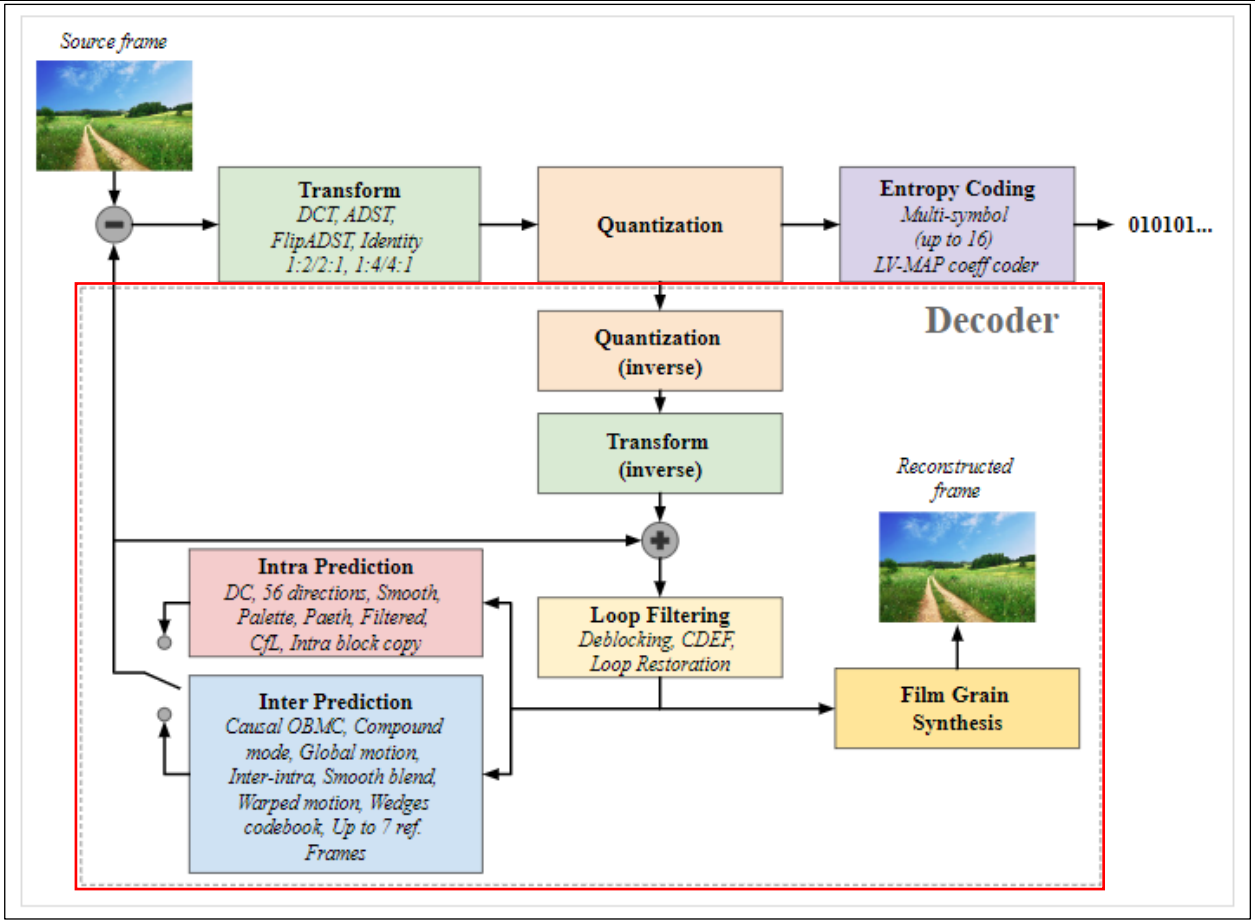
Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

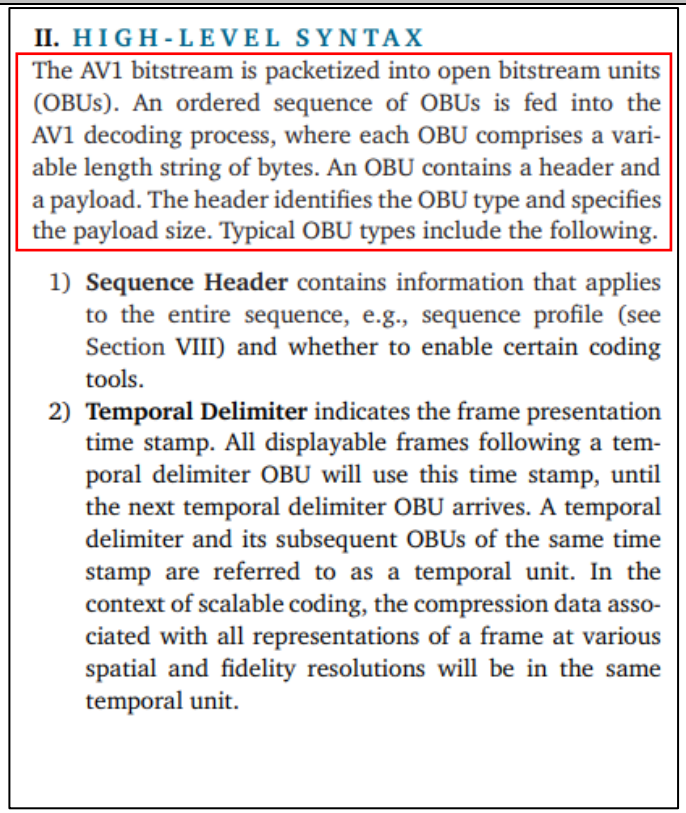
https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

The AV1 decoder reproduces reconstructed frames.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3A] a step of implementing a process for a prescribed demultiplexing on a sequence of input encoded bits, outputting at least a first and a second sequence of encoded bits;	Users of the Samsung Accused Products perform a step of implementing a process for a prescribed demultiplexing on a sequence of input encoded bits, outputting at least a first and a second sequence of encoded bits. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A includes a demultiplexer configured to work on a sequence of input encoded bits (*e.g.*, "[t]he AV1 bitstream"). The AV1 bitstream that is input into the demultiplexer "is packetized into open bitstream units (OBUs)," which are at least a first and a second sequence of encoded bits.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.

https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Samsung's 8K Neo QLED QN800A demultiplexer is configured to implement a process for a prescribed demultiplexing to output at least a first sequence of encoded bits (*e.g.*, sequence of intraframe type bits) and a second sequence of encoded bits (*e.g.*, sequence of interframe type bits). The information regarding the type of bit (and, consequently, whether it falls within the first or second sequence of encoded bits) is located in the

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	frame headers of the various Open Bitstream Units (OBUs), "which provides identifying information for the contained data (payload)." **OBU** All structures are packetized in "Open Bitstream Units" or OBUs. Each OBU has a header, which provides identifying information for the contained data (payload). https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method. 4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering. 5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost. 6) **Metadata** carries information, such as high dynamic range, scalability, and timecode. 7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13]. We refer to [9] for bit field definitions and more detailed consideration of high-level syntax. https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

5.9. Frame header OBU syntax

5.9.2. Uncompressed header syntax

uncompressed_header( ) {	Type
if (frame_id_numbers_present_flag) {	
idLen = (additional_frame_id_length_minus_1 +	
delta_frame_id_length_minus_2 + 3)	
}	

allFrames = (1 << NUM_REF_FRAMES) - 1	
if (reduced_still_picture_header) {	
show_existing_frame = 0	
frame_type = KEY_FRAME	
FrameIsIntra = 1	
show_frame = 1	

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

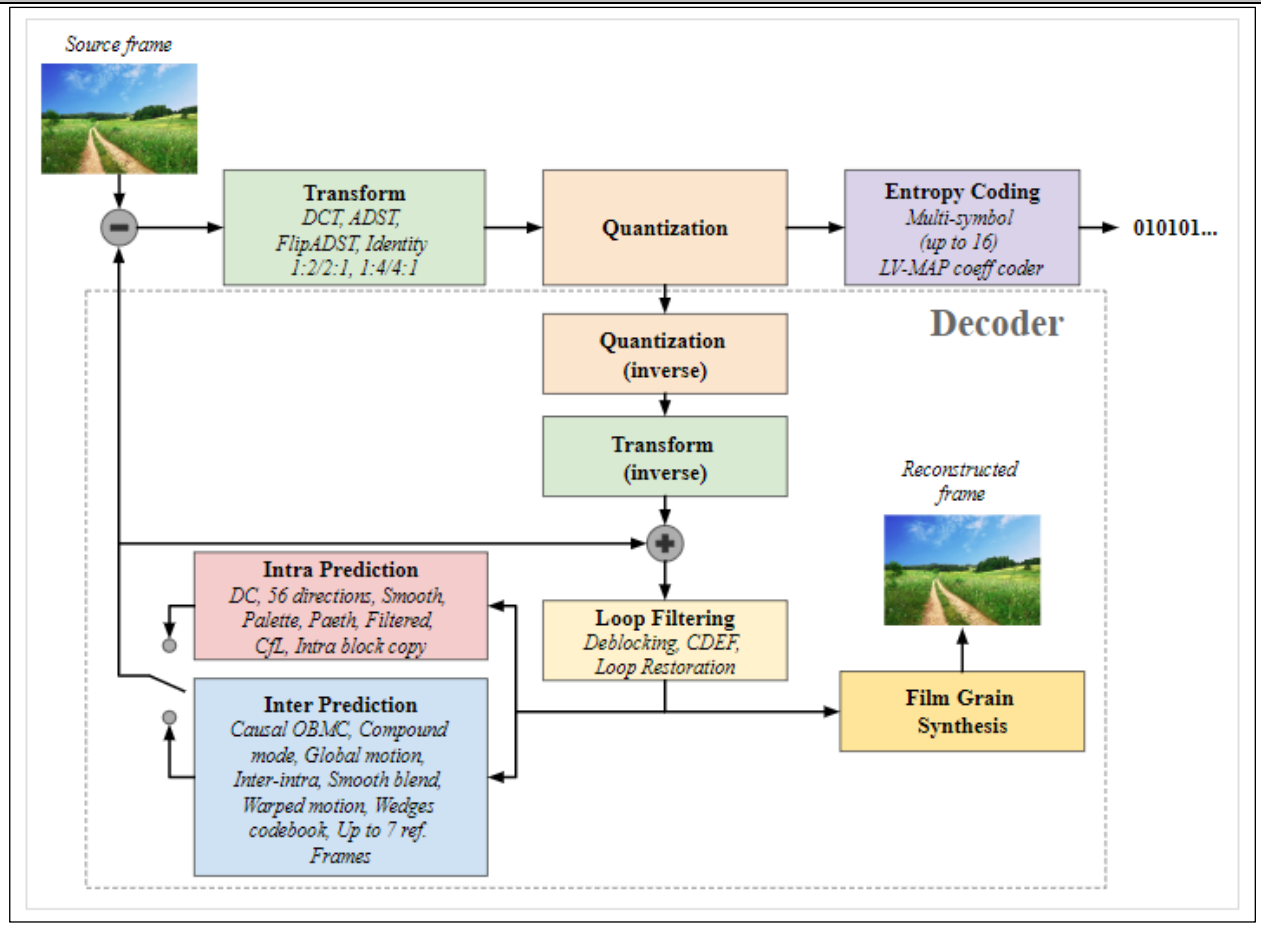
6.8.2. Uncompressed header semantics

frame_type specifies the type of the frame:

frame_type	Name of frame_type
0	KEY_FRAME
1	INTER_FRAME
2	INTRA_ONLY_FRAME
3	SWITCH_FRAME

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Inter frame A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Intra frame** A frame compressed using only intra prediction which can be independently decoded. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3B] a step of acquiring the first sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing, implementing thereon a process for a	Users of the Samsung Accused Products perform a step of acquiring the first sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing, implementing thereon a process for a prescribed first decoding, creating a sequence of decoded pictures with the standard resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses the AV1 codec, which comprises a first decoder configured to acquire the first sequence of encoded bits obtained with a standard resolution (*e.g.*, sequence of intraframes) at the demultiplexer to implement thereon a process for a prescribed first decoding to create a sequence of decoded pictures with a standard resolution (*e.g.*, a sequence of decoded intraframes).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
prescribed first decoding, creating a sequence of decoded pictures with the standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, the sequence of decoded pictures with a standard resolution can be super-resolved or non-super-resolved depending on the use_superres bit information sent within each frame header. When use_superres is equal to 1, upscaling is needed. When use_superres is not equal to 1, the first decoder is configured to implement a process that creates a sequence of decoded pictures where upscaling is not needed (*i.e.*, a sequence of decoded pictures with a standard resolution). 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3C] a step of acquiring the sequence of decoded pictures created with the standard	Users of the Samsung Accused Products perform a step of acquiring the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding, working on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement, creating a sequence of super-resolution enlarged decoded pictures with a first resolution higher than the standard resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
resolution through the process for the prescribed first decoding, working on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement, creating a sequence of super-resolution enlarged decoded pictures with a first resolution higher than the standard resolution;	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a first super-resolution enlarger configured to (for intraframe sequences for which super-resolution is active) acquire the sequence of decoded pictures created with a standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframe bits) to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement (*e.g.*, through upscaling and loop restoration) to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution. As described above with respect to Claim [2B] and replicated below, when use_superres is equal to 1, upscaling (*i.e.*, a first super-resolution) is needed for the sequence of decoded pictures (*e.g.*, pictures based on the intraframe bit sequence at standard resolution). **6.8.7. Superres params semantics** **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf As noted below, the AV1 codec utilizes a "frame super-resolution framework" that comprises two steps: (1) "Linear upscale" and (2) "Loop restore".

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

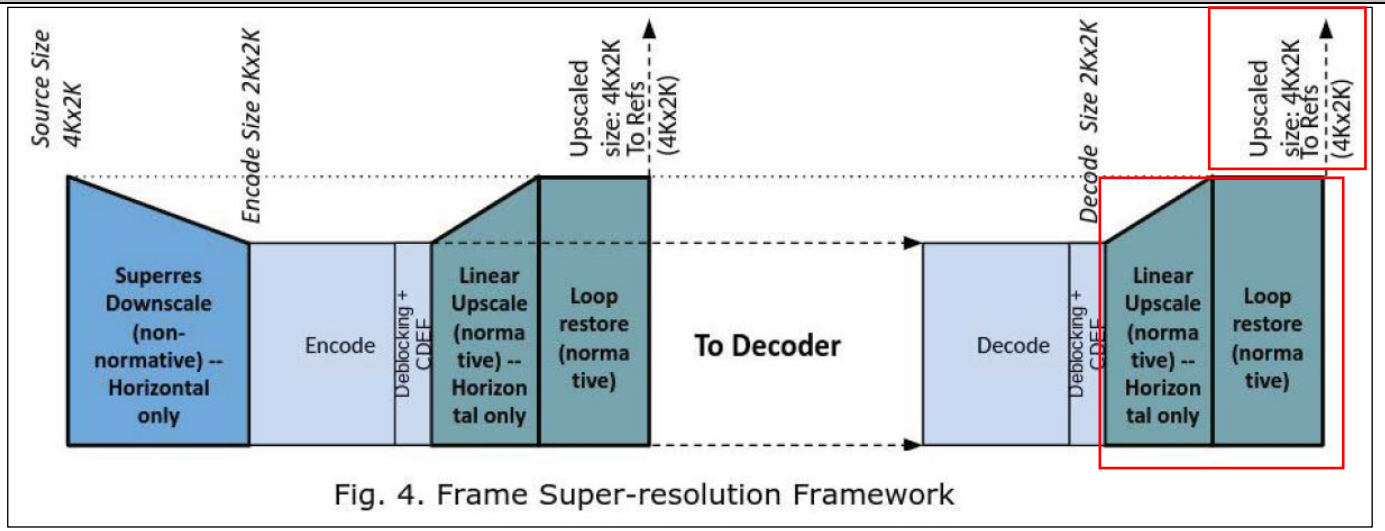


Fig. 4. Frame Super-resolution Framework

https://ieeexplore.ieee.org/document/8954553

7.16. Upscaling process

Input to this process is an array inputFrame of width FrameWidth and height FrameHeight.

The output of this process is a horizontally upscaled frame of width UpscaledWidth and height FrameHeight.

If use_superres is equal to 0, no upscaling is required and this process returns inputFrame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

This process is specified as:

```
for ( plane = 0; plane < NumPlanes; plane++ ) {
    if ( plane > 0 ) {
        subX = subsampling_x
        subY = subsampling_y
    } else {
        subX = 0
        subY = 0
    }
    downscaledPlaneW = Round2(FrameWidth, subX)
    upscaledPlaneW = Round2(UpscaledWidth, subX)
    planeH = Round2(FrameHeight, subY)
    stepX = ((downscaledPlaneW << SUPERRES_SCALE_BITS) + (upscaledPlaneW / 2)) / upscaledPlaneW
    err = (upscaledPlaneW * stepX) - (downscaledPlaneW << SUPERRES_SCALE_BITS)
    initialSubpelX =
      (-((upscaledPlaneW - downscaledPlaneW) << (SUPERRES_SCALE_BITS - 1)) + upscaledPlaneW / 2) /
upscaledPlaneW +
      (1 << (SUPERRES_EXTRA_BITS - 1)) - err / 2
    initialSubpelX &= SUPERRES_SCALE_MASK
    miW = MiCols >> subX
    minX = 0
    maxX = miW * MI_SIZE - 1
    for ( y = 0; y < planeH; y++ ) {
        for ( x = 0; x < upscaledPlaneW; x++ ) {
            srcX = -(1 << SUPERRES_SCALE_BITS) + initialSubpelX + x*stepX
            srcXPx = (srcX >> SUPERRES_SCALE_BITS)
            srcXSubpel = (srcX & SUPERRES_SCALE_MASK) >> SUPERRES_EXTRA_BITS
            sum = 0
            for ( k = 0; k < SUPERRES_FILTER_TAPS; k++ ) {
                sampleX = Clip3(minX, maxX, srcXPx + (k - SUPERRES_FILTER_OFFSET))
                px = frame[plane][y][sampleX]
                sum += px * Upscale_Filter[srcXSubpel][k]
            }
            outputFrame[plane][y][x] = Clip1(Round2(sum, FILTER_BITS))
        }
    }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17. Loop restoration process

Input to this process are the arrays UpscaledCurrFrame (of reconstructed samples) and UpscaledCdefFrame (of deringed samples).

Output from this process is the array LrFrame of loop restored samples.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

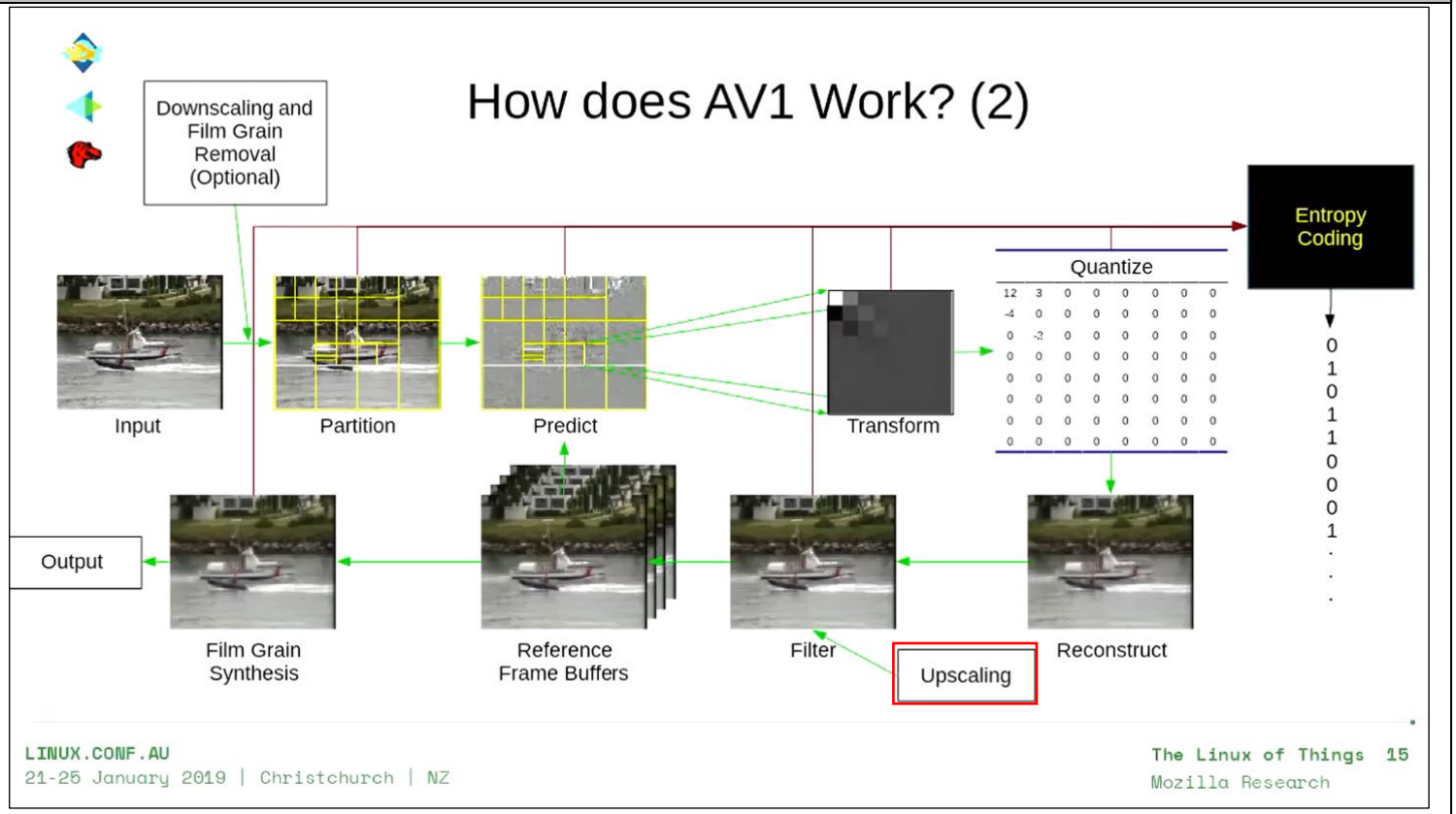
The array LrFrame is set equal to a copy of UpscaledCdefFrame. (The contents of LrFrame will later be overwritten for blocks that require restoration filtering.)

If UsesLr is equal to 0, then the process returns immediately after performing this copy.

Otherwise, loop restoration is applied as follows:

```
for ( y = 0; y < FrameHeight; y += MI_SIZE ) {
  for ( x = 0; x < UpscaledWidth; x += MI_SIZE ) {
    for ( plane = 0; plane < NumPlanes; plane++ ) {
      if ( FrameRestorationType[ plane ] != RESTORE_NONE ) {
        row = y >> MI_SIZE_LOG2
        col = x >> MI_SIZE_LOG2
        loop_restore_block( plane, row, col )
      }
    }
  }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, an interpolation of pixels is applied on the decoded intraframe pictures to create a super resolution enlarged decoded picture. In AV1, the interpolation process is performed by wiener filter process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17.1. Loop restore block process

The inputs to this process are:

- a variable plane specifying whether the process is filtering Y, U, or V samples,
- variables row and col specifying the location of the block in units of 4x4 blocks in the upscaled luma plane.

The output of this process are samples in LrFrame[plane].

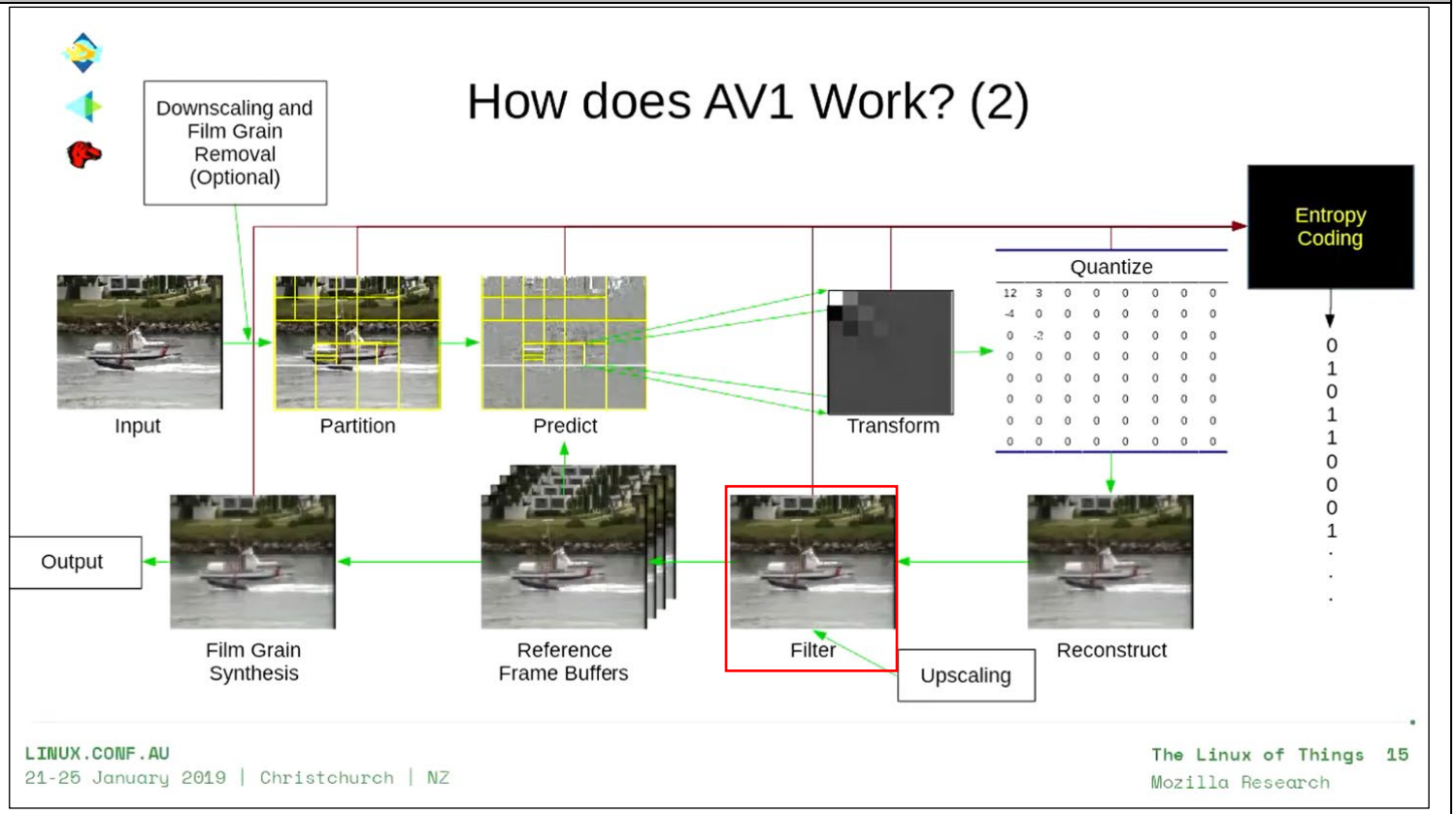
The filter to used depends on rType as follows:

- If rType is equal to RESTORE_WIENER, the Wiener filter process specified in section 7.17.4 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise, if rType is equal to RESTORE_SGRPROJ, the self guided filter process specified in section 7.17.2 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise (rType is equal to RESTORE_NONE), no filtering is applied.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	7.17.4. Wiener filter process The inputs to this block are: • a variable plane specifying whether the process is filtering Y, U, or V samples, • variables unitRow and unitCol specifying the position of the loop restoration unit, • variables x and y specifying the position of the block in samples relative to the top-left corner of the current plane, • variables w and h specifying the size of the block in samples. The output from this process are modified samples in LrFrame. The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. https://aomediacodec.github.io/av1-spec/av1-spec.pdf

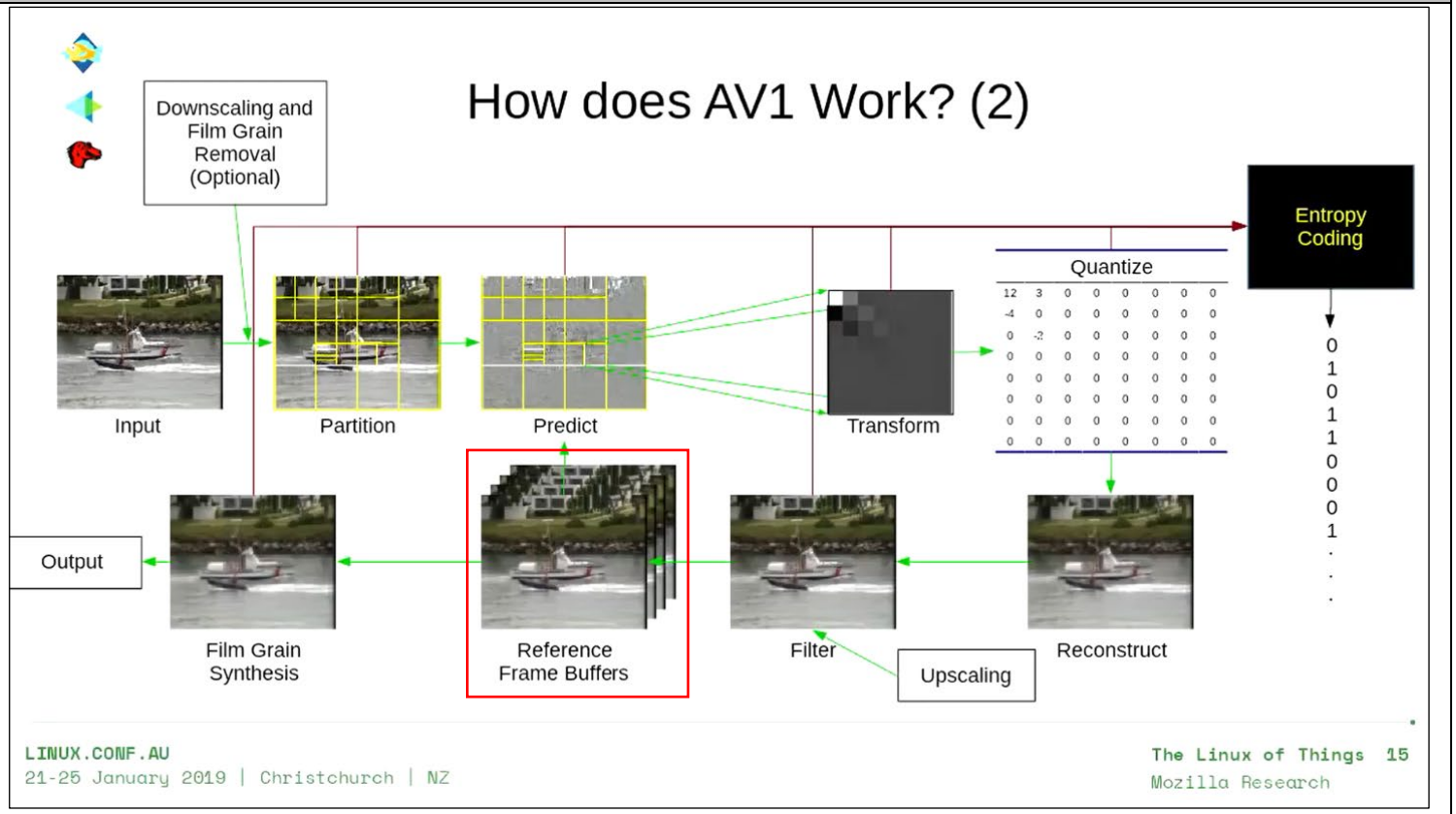
Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[3D] a step of acquiring the sequence of super-resolution enlarged decoded pictures created through the process for the prescribed super-resolution enlargement working on the sequence of super-resolution enlarged decoded pictures to implement a process for a first resolution conversion, creating a sequence of super-resolution decoded pictures with a	Users of the Samsung Accused Products perform a step of acquiring the sequence of super-resolution enlarged decoded pictures created through the process for the prescribed super-resolution enlargement working on the sequence of super-resolution enlarged decoded pictures to implement a process for a first resolution conversion, creating a sequence of super-resolution decoded pictures with a standard resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger. After the loop restoration process, the reconstructed intraframes are added to the reference buffer list which are further used for decoding interframes. The references pictures at the decoding side are scaled according to the resolution of the current interframe to be decoded. Since the first super-resolution enlarger provides an upscaled decoded reference picture, the reference pictures are downscaled to match the current interframe's resolution to be used as a reference picture for those interframes.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

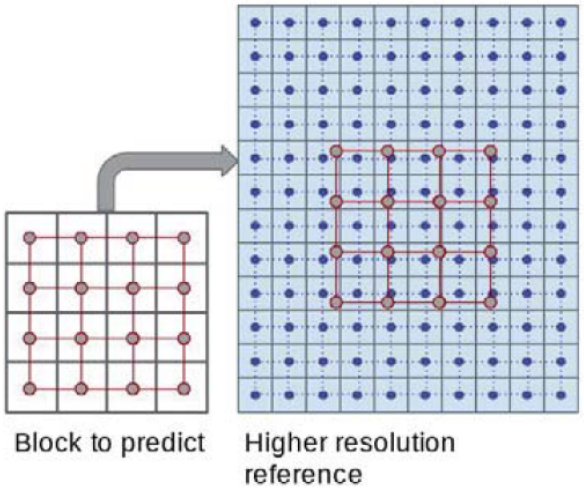
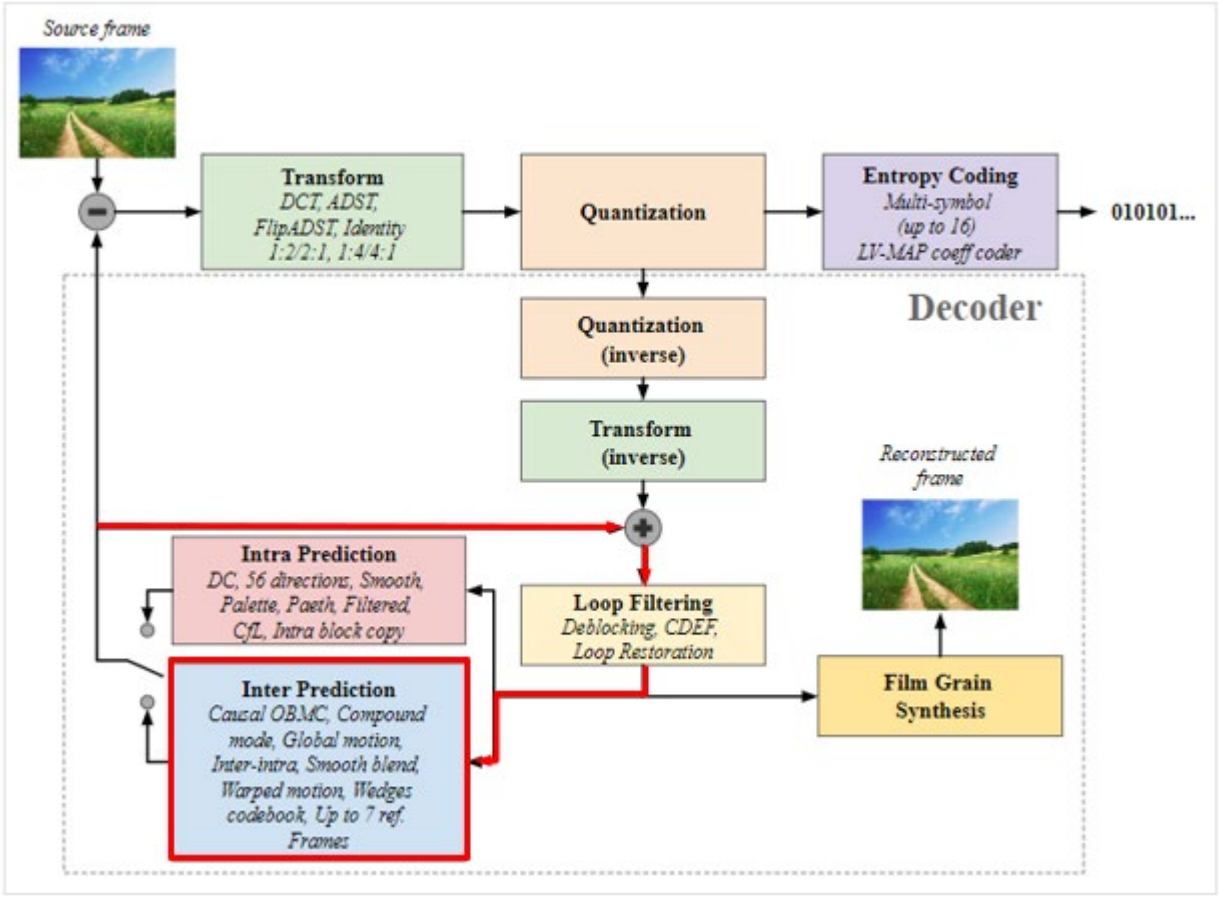


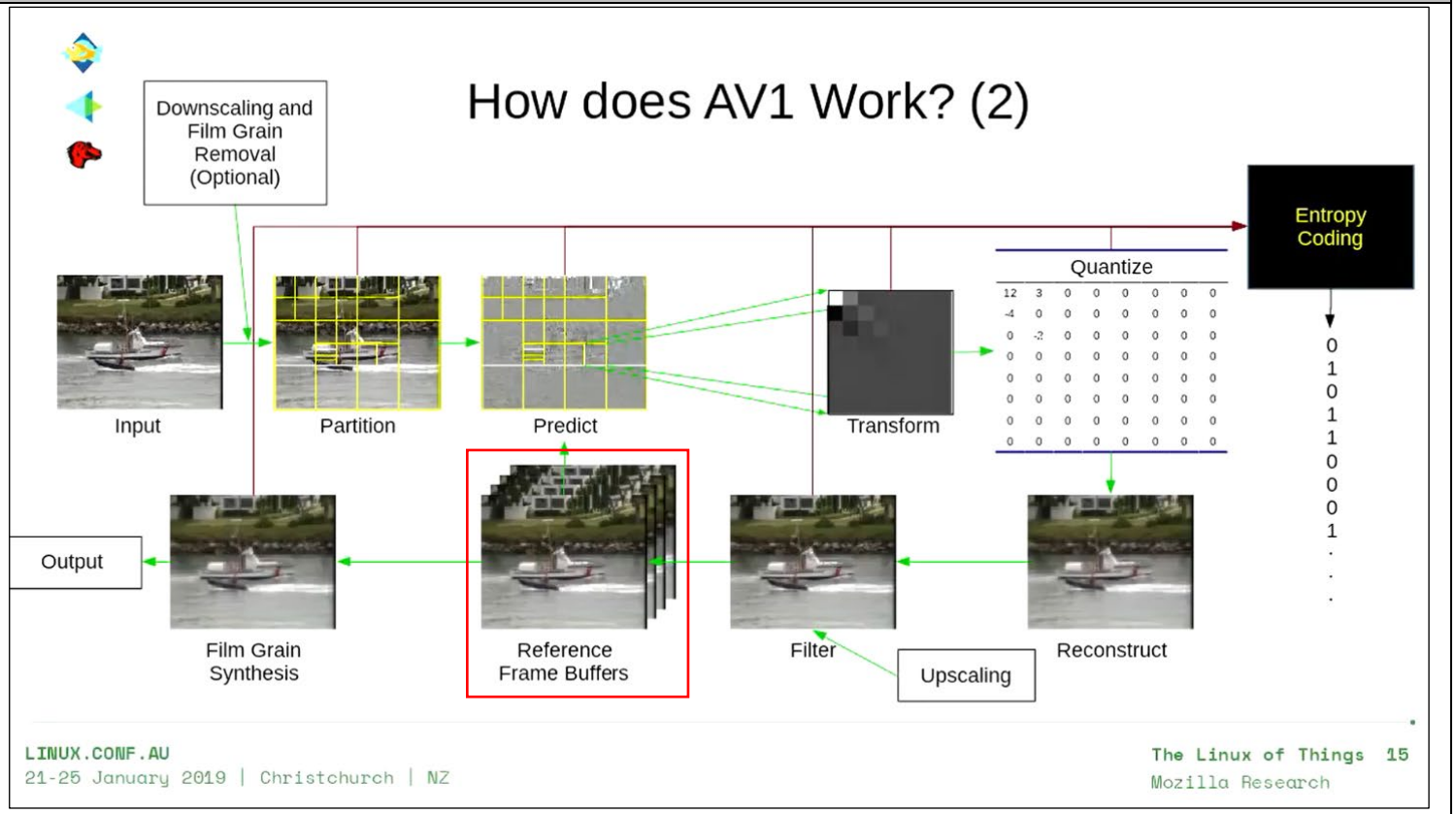
Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3E] a step of acquiring the second sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing as a set of decoding targets, the sequence of decoded pictures created with the standard resolution through the process for the	Users of the Samsung Accused Products perform a step of acquiring the second sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing as a set of decoding targets, the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution through the process for the prescribed resolution conversion as a set of second reference pictures. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer (*e.g.*, sequence of interframes) as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframes) as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures (*e.g.*, intraframes that are downscaled to match the current interframe's resolution to be used as a reference picture). As described above with respect to Claim Element [3A], the demultiplexer outputs two sequences of encoded bits, where the second sequence of encoded bits is the sequence of interframe bits. This sequence of interframe bits is a set of decoding targets because reconstructed interframes without super-resolution active (*i.e.*, the second sequence of encoded bits obtained with a standard resolution) are used for inter-prediction of the current frame in the decoding process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
prescribed first decoding as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution through the process for the prescribed resolution conversion as a set of second reference pictures, and	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, as described above with respect to Claim Element [3B], the first decoder creates a sequence of decoded pictures (based on the first sequence of intraframe bits) created with the standard resolution at the first decoder

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	as a set of first reference pictures since these intraframes create the reference frames for the inter prediction described above. 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM.[7] https://aomediacodec.github.io/av1-spec/av1-spec.pdf Moreover, as described above with respect to Claim Elements [3C] and [3D], a first resolution converter converts the sequence of super-resolution decoded pictures with the standard resolution. The AV1 decoder pipeline, with super-resolution active, produces decoded frames which are references that are super-resolved and then downscaled to match the current frame resolution (*i.e.*, the set of super-resolution enlarged and converted decoded pictures from the second resolution converter that acts as a set of second reference pictures).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

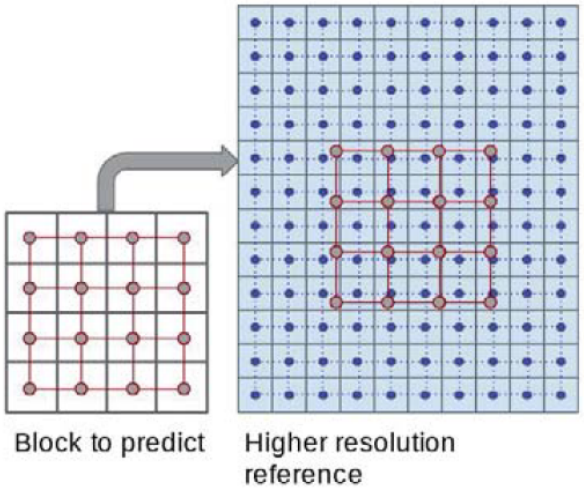
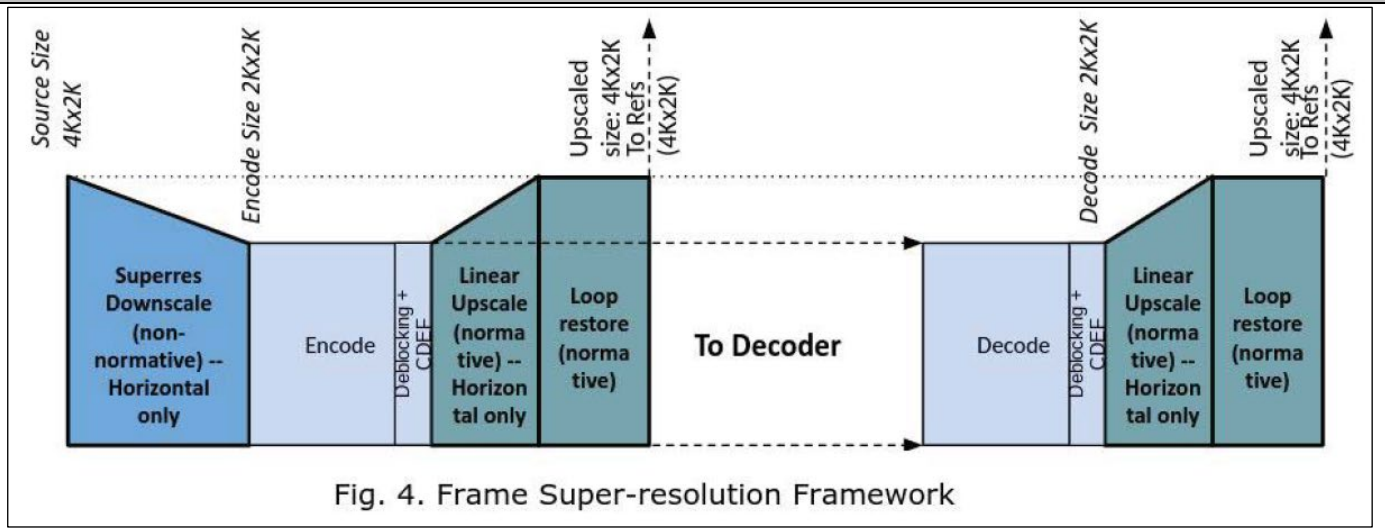


Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Therefore, both the sequence of decoded pictures created initially at standard resolution and the sequence of decoded pictures created by first upscaling and then downscaling, are sets of reference pictures acquired by the second decoder along with the second sequence comprising interframe pictures to perform the requisite interframe prediction. *B.1 Framework At a Glance* Fig. 4 depicts the overall in-loop filtering pipeline in AV1 and the super-resolution framework. According to this framework, on the encoder side, a source frame may be first downscaled non-normatively and encoded at a lower resolution. The deblocking filter, and another filter called Constrained Directional Enhancement Filter (CDEF) [17] supported in AV1, are then used to remove blocking and ringing artifacts while preserving edges, at the lower resolution. This is followed by a normative Linear Upsampling

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	process to full source resolution. Finally, another AV1 tool called Loop Restoration (LR) [18] is applied at full resolution to recover the high frequencies lost during the prior downscaling and quantization processes. The loop restored frame updates the reference buffer store for encoding subsequent frames. A similar process is followed on the decoder side as well. Note that the Deblocking, CDEF, Linear Upscaling and LR in order are all part of the normative in-loop filtering process in AV1, where each filter can be turned on or off per frame. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Fig. 4. Frame Super-resolution Framework https://ieeexplore.ieee.org/document/8954553 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3F] a step of... selecting one of the set of first reference pictures and the set of second reference pictures based	Users of the Samsung Accused Products perform a step of selecting one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, based on the set of decoding targets and the set of selected reference pictures to create a sequence of super-resolution pictures decoded with the standard resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, based on the set of decoding targets and the set of selected reference pictures to create a sequence of super-resolution pictures decoded with the standard resolution; and	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a second decoder configured to select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information (*e.g.*, selected based on a reference index) to implement a combination of processes for a prescribed prediction (*e.g.*, reconstructed frames without super-resolution are used for inter prediction of the current frame) and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution (*e.g.*, downscaled) based on the set of decoding targets (*e.g.*, resolution of the current frame) and the set of selected reference pictures (*e.g.*, the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index). As shown below, the AV1 codec supports the ability to predict pictures across scales using upscalers and downscalers (*i.e.*, a reference picture at a first resolution can be used to predict a picture at a second resolution). The second decoder waits for the current interframe to be decoded as received from the demultiplexer. When it receives the Interframe, the frame can be decoded based on the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

https://ieeexplore.ieee.org/document/8954553

Since AV1 specification allow each frame to either be normally decoded or decoded with super-resolve steps, it is implied that the reference picture buffer set consists of both non-super-resolved and super-resolved intraframes that can be used as reference pictures. For the second decoder to decode the current interframe, the reference frame is selected based on the reference index. Thus, the reference index that indicates which reconstructed frame is selected for prediction is implied as 'the reference picture selection information' that is sent in the encoded bitstream.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	ref_frame_idx[i] specifies which reference frames are used by inter frames. It is a requirement of bitstream conformance that RefValid[ref_frame_idx[i]] is equal to 1, and that the selected reference frames match the current frame in bit depth, profile, chroma subsampling, and color space. **Note:** Syntax elements indicate a reference (such as LAST_FRAME, ALTREF_FRAME). These references are looked up in the ref_frame_idx array to find which reference frame should be used during inter prediction. There is no requirement that the values in ref_frame_idx should be distinct. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3G] a step of acquiring the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding working on the	Users of the Samsung Accused Products perform a step of acquiring the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding working on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement, creating a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A comprises a second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder (*e.g.*, a sequence of decoded intraframes) to work on the sequence of decoded pictures to implement an interpolation of pixels with

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
sequence of decoded pictures to implement an interpolation of pixels with the second enlargement, creating a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution,	the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution (*e.g.*, a sequence of upscaled decoded intraframes). Different 8-tap filter coefficients can be used for upscaling of the decoded frame. Google Git aomedia / aom / refs/heads/main / . / av1 / common / **resize.c** blob: f4bfcd062af2a6c4643314ec1501aa9c27306aa2 [file] [log] [blame] 1 /* 2 * Copyright (c) 2016, Alliance for Open Media. All rights reserved 3 * 4 * This source code is subject to the terms of the BSD 2 Clause License and 5 * the Alliance for Open Media Patent License 1.0. If the BSD 2 Clause License 6 * was not distributed with this source code in the LICENSE file, you can 7 * obtain it at www.aomedia.org/license/software. If the Alliance for Open 8 * Media Patent License 1.0 was not distributed with this source code in the 9 * PATENTS file, you can obtain it at www.aomedia.org/license/patent. 10 */

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

```
176  const int16_t av1_resize_filter_normative[(
177      1 << RS_SUBPEL_BITS)][UPSCALE_NORMATIVE_TAPS] = {
178  #if UPSCALE_NORMATIVE_TAPS == 8
179    { 0, 0, 0, 128, 0, 0, 0, 0 },        { 0, 0, -1, 128, 2, -1, 0, 0 },
180    { 0, 1, -3, 127, 4, -2, 1, 0 },      { 0, 1, -4, 127, 6, -3, 1, 0 },
181    { 0, 2, -6, 126, 8, -3, 1, 0 },      { 0, 2, -7, 125, 11, -4, 1, 0 },
182    { -1, 2, -8, 125, 13, -5, 2, 0 },    { -1, 3, -9, 124, 15, -6, 2, 0 },
```

https://aomedia.googlesource.com/aom/+/refs/heads/main/av1/common/resize.c

The AV1 codec accomplishes prediction across frames of different resolutions through its normative upscaling filters.

For example, intraframes or Alternate Reference Frames, which are normally decoded by the first decoder, may require upscaling to match resolution of current frame. Since the AV1 codec allows switching between resolutions and using references of different resolutions, and coding can be done in an arbitrary resolution, a reconstructed frame which is to be used as reference might be present in a reconstructed-reference buffer whose resolution is different (lower) than the resolution of the interframe currently being decoded, for which the intraframe is an indexed reference. In such a scenario, a resolution conversion (*e.g.*, an upscale) would be required for decoding the current interframe. Such a resolution upscale is the AV1 codec's second resolution converter.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

https://ieeexplore.ieee.org/document/8954553

Note that the AV1 bit-stream is flexible enough that each coded frame can use (a) horizontal-only super-resolution mode as described above, (b) code a frame at an arbitrary resolution within the ratio constraints, (c) a combination of both, or (d) no scaling at all. An intelligent encoder can utilize these options as additional knobs to be

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[3H] wherein the set of decoding targets, the set of first reference pictures, and the set of second reference pictures have the same value in spatial resolution.	The set of decoding target pictures, the set of first reference pictures, and the set of second reference pictures in the Samsung Accused Products have the same value in spatial resolution. Samsung directly infringes and induces infringement by causing a user to perform the claimed method on devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A upscales or downscales the reference pictures to match the resolution of the decoding targets. That is, the AV1 codec in the Samsung 8K Neo QLED QN800A scales (1) the set of decoding target pictures; (2) the set of first reference pictures, and (3) the set of second reference pictures to have the same value in spatial resolution. As described below, "the motion vectors transmitted in the bitstream are always at the source resolution."

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

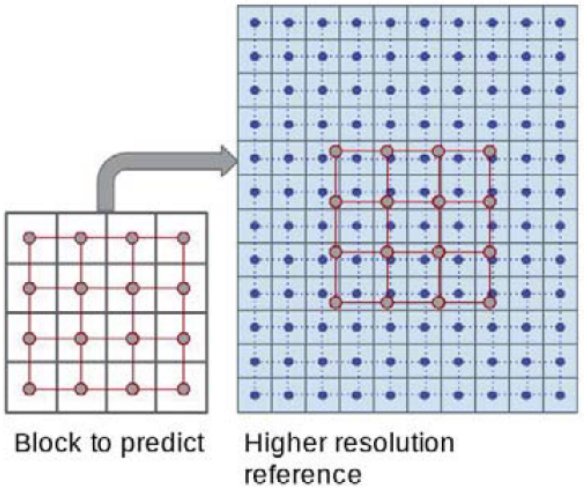


Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4P] A recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute:	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A includes a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium that that is configured to have a computer execute moving picture decoding using the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Compatible Samsung TVs

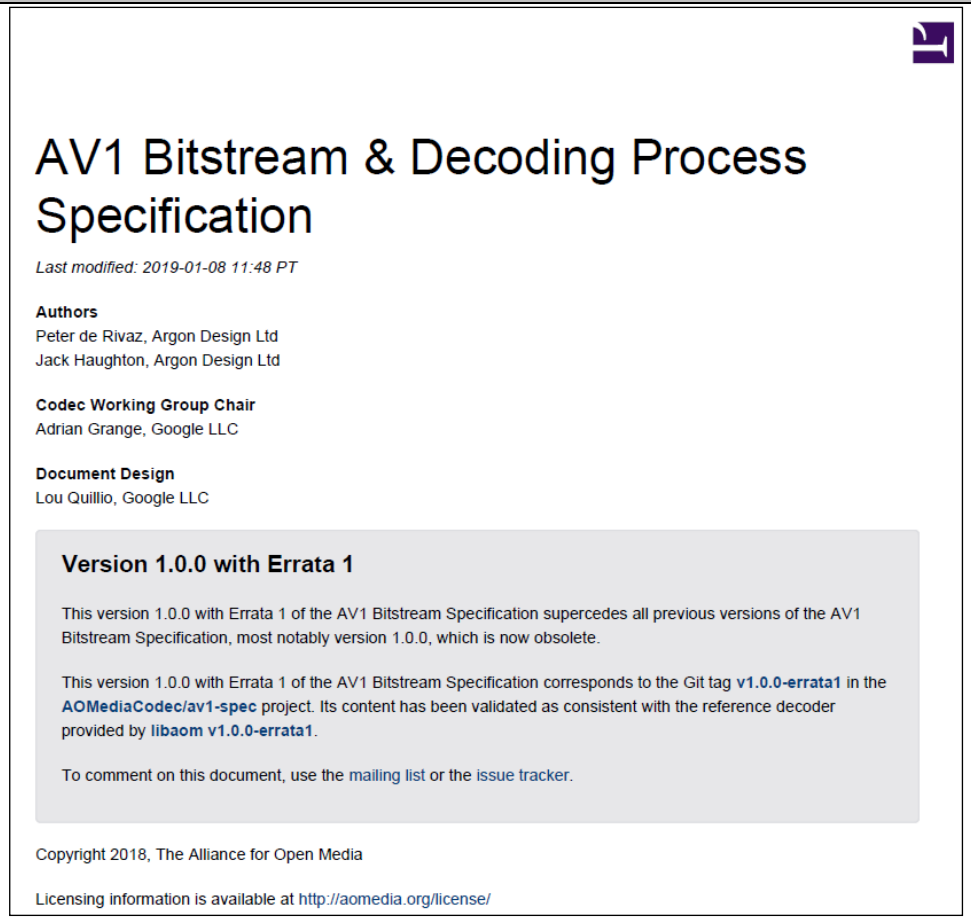
Samsung has been adding AV1 support to its televisions, and currently, around 35 models launched in 2020 and 2021 support the open-source codec.

TV Series	Models
Neo QLED 8K	QN900A, QN800A, QN700A
QLED 8K	Q950T, Q900T, Q800T, Q700T
Neo QLED 4K	QN95A, QN91A, QN90A, QN85A
QLED 4K	Q80A, Q70A, Q60A, Q95T, Q90T, Q80T, Q70T, Q60T
Crystal UHD	AU9000, AU8000, AU7100, TU8500, TU8300, TU8000, TU7100, TU7000
The Serif	LS01T
The Frame	LS03A, LS03T

https://sammyhub.com/2021/11/19/samsung-tvs-with-av1-codec-support/

The AV1 codec is used for video decoding, which is moving picture decoding.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



AV1 Bitstream & Decoding Process Specification

Last modified: 2019-01-08 11:48 PT

Authors
Peter de Rivaz, Argon Design Ltd
Jack Haughton, Argon Design Ltd

Codec Working Group Chair
Adrian Grange, Google LLC

Document Design
Lou Quillio, Google LLC

Version 1.0.0 with Errata 1

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification supercedes all previous versions of the AV1 Bitstream Specification, most notably version 1.0.0, which is now obsolete.

This version 1.0.0 with Errata 1 of the AV1 Bitstream Specification corresponds to the Git tag **v1.0.0-errata1** in the **AOMediaCodec/av1-spec** project. Its content has been validated as consistent with the reference decoder provided by **libaom v1.0.0-errata1**.

To comment on this document, use the mailing list or the issue tracker.

Copyright 2018, The Alliance for Open Media

Licensing information is available at http://aomedia.org/license/

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

The Samsung 8K Neo QLED QN800A television supports the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

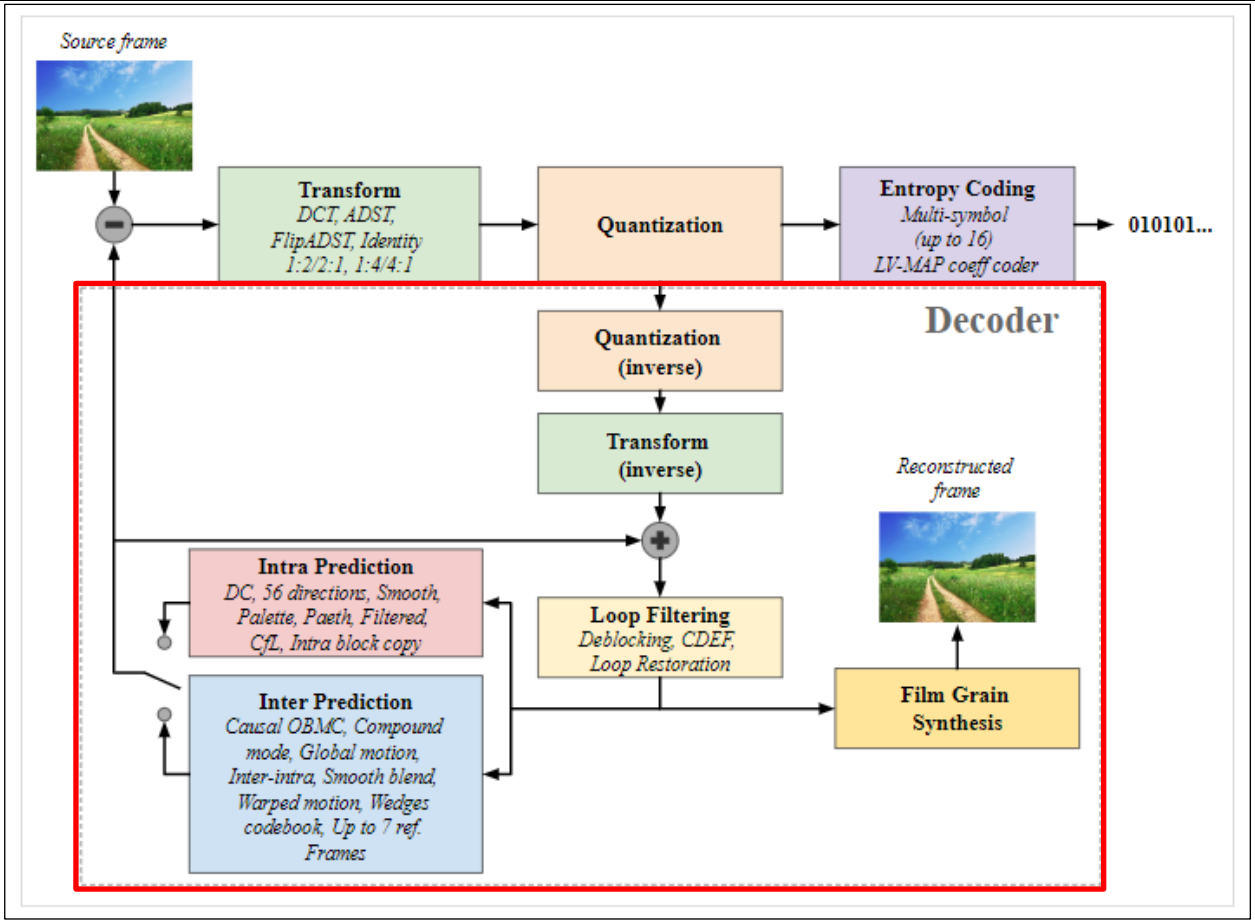
Supported video codecs (QN7**A Series or higher)

File format	Container	Video codecs	Resolution	Frame rate (fps)	Bitrate (Mbps)	Audio codecs
*.avi *.mkv *.asf *.wmv *.mp4 *.mov *.3gp *.vro *.mpg *.mpeg *.ts *.tp *.trp *.flv *.vob *.svi *.m2ts *.mts	AVI MKV ASF MP4 3GP MOV FLV VRO VOB PS TS SVAF	H.264 BP/MP/HP	3840 x 2160	3840 x 2160: 60 1920 x 1080: 120	60	Dolby Digital LPCM ADPCM(IMA, MS) AAC HE-AAC WMA Dolby Digital+ MPEG(MP3) AC-4 G.711(A-Law, μ-Law) OPUS
		HEVC (H.265 - Main, Main10)	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	100	
		Motion JPEG	3840 x 2160	30	80	
		MVC	1920 x 1080	60	20	
		MPEG4 SP/ASP				
		Window Media Video v9 (VC1)				
		MPEG2				
		MPEG1				
		Microsoft MPEG-4 v1, v2, v3		30		
		Window Media Video v7 (WMV1), v8 (WMV2)				
		H.263 Sorrenson				
		VP6				
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	
*.webm	WebM	VP8	1920 x 1080	60	20	Vorbis
		VP9 (Profile 0, profile 2 supported)	3840 x 2160	60	80	
		AV1	7680 x 4320	7680 x 4320: 60 3840 x 2160: 120	80	

https://downloadcenter.samsung.com/content/UM/202208/20220802173319219/OSNDVBADA-7.2.0_EM_OSCAR_ASIA_ENG_220110.0.pdf

The AV1 decoder reproduces reconstructed frames.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products



https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Moreover, the AV1 codec comprises a non-transitory computer-readable recording medium for storing its moving picture decoding program. The recording medium is a combination of two forms of memory: DRAM

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	(dynamic random access memory) and SRAM (static random access memory) with a bus in between the two memories. As an example of these computer-readable memories configured to have a computer execute the AV1 codec's moving picture decoding program (i.e., AV1 video compression), reference frame motion information needs to be transferred to the SRAM from the DRAM "for computing purposes."

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

***Abstract*—The AV1 video compression format is developed by the Alliance for Open Media consortium. It achieves more than 30% reduction in bit-rate compared to its predecessor VP9 for the same decoded video quality. This paper provides a technical overview of the AV1 codec design that enables the compression performance gains with considerations for hardware feasibility.**

***Index Terms*—AV1, Alliance of Open Media, video compression**

I. Introduction

THE last decade has seen a steady and significant growth of web-based video applications including video-on-demand (VoD) service, live streaming, conferencing, and virtual reality [1]. Bandwidth and storage costs have driven the need for video compression techniques with better compression efficiency. VP9 [2] and HEVC [3], both debuted in 2013, achieved in the range of 50% higher compression performance [4] than the prior codec H.264/AVC [5] and were quickly adopted by the industry.

As the demand for high performance video compression continued to grow, the Alliance for Open Media [6] was formed in 2015 as a consortium for the development of open, royalty-free technology for multimedia delivery. Its first video compression format AV1, released in 2018, enabled about 30% compression gains over its predecessor VP9. The AV1 format is already supported by many web platforms including Android, Chrome, Microsoft Edge, and Firefox and multiple web-based video service providers, including YouTube, Netflix, Vimeo, and Bitmovin, have begun rolling out AV1 streaming services at scale.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	3) Hardware Constraints: The motion information, including the motion vector and the reference frame index, needs to be stored for later frames to build their motion fields. To reduce the memory footprint, the motion information is stored in units of 8×8 blocks. If a coding block is using compound modes, only the first motion vector is saved. The reference frame motion information is commonly stored in the dynamic random access memory (DRAM), a relatively cheaper and slower unit as compared to SRAM, in hardware decoders. It needs, however, to be transferred to SRAM for computing purposes. The bus between DRAM and SRAM is typically 32-bit wide. To facilitate efficient data transfer, a number of data format constraints are employed. We limit the codec to use motion information from up to 4 reference frames (out of 7 available frames) to build the motion field. Therefore only 2 bits are needed for the reference frame index. Furthermore, a motion vector with any component magnitude above 2^{12} will be discarded. As a result, the motion vector and reference frame index together can be represented by a 32-bit unit. https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200
[4A] a step of implementing a process for a prescribed demultiplexing on a sequence of input encoded bits, outputting at least a first	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of implementing a process for a prescribed demultiplexing on a sequence of input encoded bits, outputting at least a first and a second sequence of encoded bits. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
and a second sequence of encoded bits;	For example, the Samsung 8K Neo QLED QN800A implements a process for demultiplexing on a sequence of input encoded bits (*e.g.*, "[t]he AV1 bitstream"). The AV1 bitstream that is input into the demultiplexer "is packetized into open bitstream units (OBUs)," which are at least a first and a second sequence of encoded bits. **II. HIGH-LEVEL SYNTAX** The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following. 1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools. 2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit. https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Samsung's 8K Neo QLED QN800A demultiplexer is configured to implement a process for a prescribed demultiplexing to output at least a first sequence of encoded bits (*e.g.*, sequence of intraframe type bits) and a second sequence of encoded bits (*e.g.*, sequence of interframe type bits). The information regarding the type of bit (and, consequently, whether it falls within the first or second sequence of encoded bits) is located in the frame headers of the various Open Bitstream Units (OBUs), "which provides identifying information for the contained data (payload)." **OBU** All structures are packetized in "Open Bitstream Units" or OBUs. Each OBU has a header, which provides identifying information for the contained data (payload). https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method. 4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering. 5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost. 6) **Metadata** carries information, such as high dynamic range, scalability, and timecode. 7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13]. We refer to [9] for bit field definitions and more detailed consideration of high-level syntax. https://www.researchgate.net/publication/349650510_A_Technical_Overview_of_AV1/fulltext/603a63784585158939d3ea6f/A-Technical-Overview-of-AV1.pdf%200

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

5.9. Frame header OBU syntax

5.9.2. Uncompressed header syntax

uncompressed_header( ) {	Type
if (frame_id_numbers_present_flag) {	
idLen = (additional_frame_id_length_minus_1 +	
delta_frame_id_length_minus_2 + 3)	
}	

allFrames = (1 << NUM_REF_FRAMES) - 1	
if (reduced_still_picture_header) {	
show_existing_frame = 0	
frame_type = KEY_FRAME	
FrameIsIntra = 1	
show_frame = 1	

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

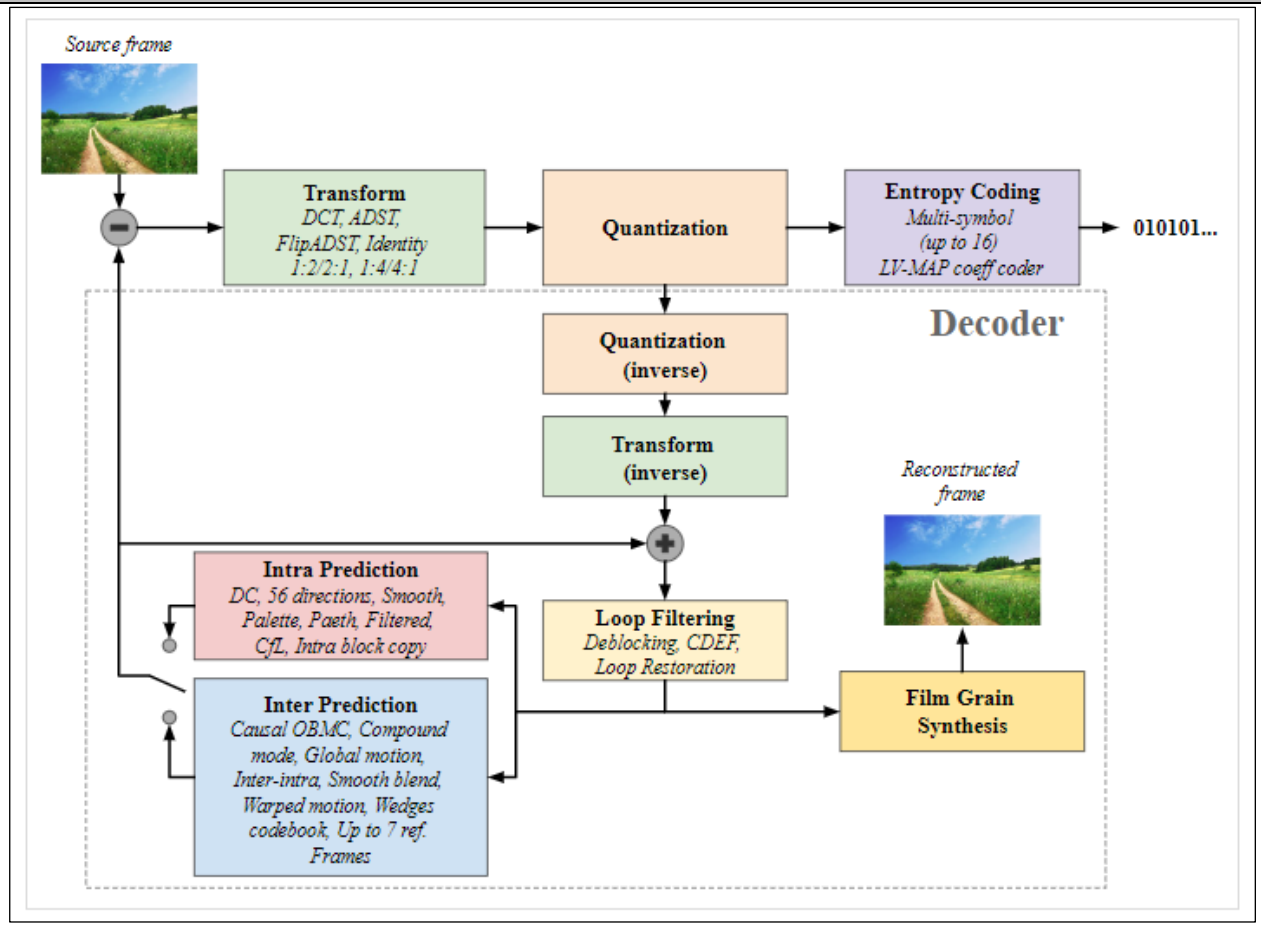
6.8.2. Uncompressed header semantics

frame_type specifies the type of the frame:

frame_type	Name of frame_type
0	KEY_FRAME
1	INTER_FRAME
2	INTRA_ONLY_FRAME
3	SWITCH_FRAME

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Inter frame A frame compressed by referencing previously decoded frames and which may use intra prediction or inter prediction. **Intra frame** A frame compressed using only intra prediction which can be independently decoded. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4B] a step of acquiring the first sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing, implementing thereon a process for a	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of acquiring the first sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing, implementing thereon a process for a prescribed first decoding, creating a sequence of decoded pictures with the standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A uses the AV1 codec, which acquires the first sequence of encoded bits obtained with a standard resolution (*e.g.*, sequence of intraframes) at the demultiplexer to implement thereon a process for a prescribed first decoding to create a sequence of decoded pictures with a standard resolution (*e.g.*, a sequence of decoded intraframes).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
prescribed first decoding, creating a sequence of decoded pictures with the standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, the sequence of decoded pictures with a standard resolution can be super-resolved or non-super-resolved depending on the use_superres bit information sent within each frame header. When use_superres is equal to 1, upscaling is needed. When use_superres is not equal to 1, the first decoder is configured to implement a process that creates a sequence of decoded pictures where upscaling is not needed (i.e., a sequence of decoded pictures with a standard resolution). 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4C] a step of acquiring the sequence of decoded pictures created with the standard	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of acquiring the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding, working on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement, creating a sequence of super-resolution enlarged decoded pictures with a first resolution higher than the standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
resolution through the process for the prescribed first decoding, working on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement, creating a sequence of super-resolution enlarged decoded pictures with a first resolution higher than the standard resolution;	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A acquires (for intraframe sequences for which super-resolution is active) the sequence of decoded pictures created with a standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframe bits) to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement (*e.g.*, through upscaling and loop restoration) to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution. As described above with respect to Claim [2B] and replicated below, when use_superres is equal to 1, upscaling (*i.e.*, a first super-resolution) is needed for the sequence of decoded pictures (*e.g.*, pictures based on the intraframe bit sequence at standard resolution). 6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM. https://aomediacodec.github.io/av1-spec/av1-spec.pdf As noted below, the AV1 codec utilizes a "frame super-resolution framework" that comprises two steps: (1) "Linear upscale" and (2) "Loop restore".

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

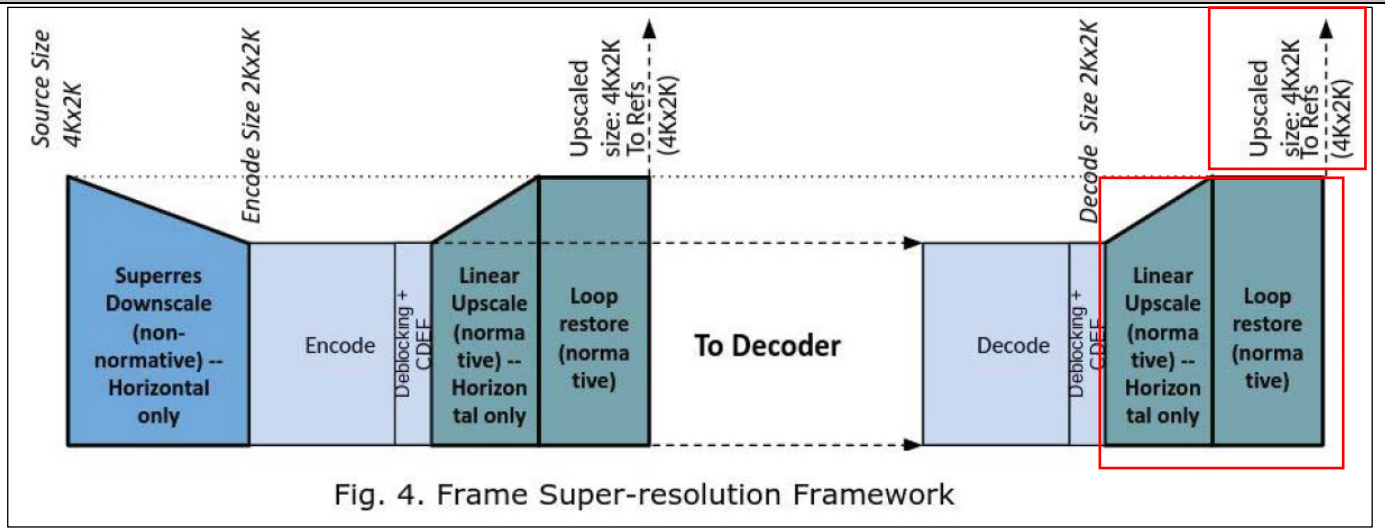


Fig. 4. Frame Super-resolution Framework

https://ieeexplore.ieee.org/document/8954553

7.16. Upscaling process

Input to this process is an array inputFrame of width FrameWidth and height FrameHeight.

The output of this process is a horizontally upscaled frame of width UpscaledWidth and height FrameHeight.

If use_superres is equal to 0, no upscaling is required and this process returns inputFrame.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

This process is specified as:

```
for ( plane = 0; plane < NumPlanes; plane++ ) {
    if ( plane > 0 ) {
        subX = subsampling_x
        subY = subsampling_y
    } else {
        subX = 0
        subY = 0
    }
    downscaledPlaneW = Round2(FrameWidth, subX)
    upscaledPlaneW = Round2(UpscaledWidth, subX)
    planeH = Round2(FrameHeight, subY)
    stepX = ((downscaledPlaneW << SUPERRES_SCALE_BITS) + (upscaledPlaneW / 2)) / upscaledPlaneW
    err = (upscaledPlaneW * stepX) - (downscaledPlaneW << SUPERRES_SCALE_BITS)
    initialSubpelX =
      (-((upscaledPlaneW - downscaledPlaneW) << (SUPERRES_SCALE_BITS - 1)) + upscaledPlaneW / 2) /
upscaledPlaneW +
      (1 << (SUPERRES_EXTRA_BITS - 1)) - err / 2
    initialSubpelX &= SUPERRES_SCALE_MASK
    miW = MiCols >> subX
    minX = 0
    maxX = miW * MI_SIZE - 1
    for ( y = 0; y < planeH; y++ ) {
        for ( x = 0; x < upscaledPlaneW; x++ ) {
            srcX = -(1 << SUPERRES_SCALE_BITS) + initialSubpelX + x*stepX
            srcXPx = (srcX >> SUPERRES_SCALE_BITS)
            srcXSubpel = (srcX & SUPERRES_SCALE_MASK) >> SUPERRES_EXTRA_BITS
            sum = 0
            for ( k = 0; k < SUPERRES_FILTER_TAPS; k++ ) {
                sampleX = Clip3(minX, maxX, srcXPx + (k - SUPERRES_FILTER_OFFSET))
                px = frame[plane][y][sampleX]
                sum += px * Upscale_Filter[srcXSubpel][k]
            }
            outputFrame[plane][y][x] = Clip1(Round2(sum, FILTER_BITS))
        }
    }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17. Loop restoration process

Input to this process are the arrays UpscaledCurrFrame (of reconstructed samples) and UpscaledCdefFrame (of deringed samples).

Output from this process is the array LrFrame of loop restored samples.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

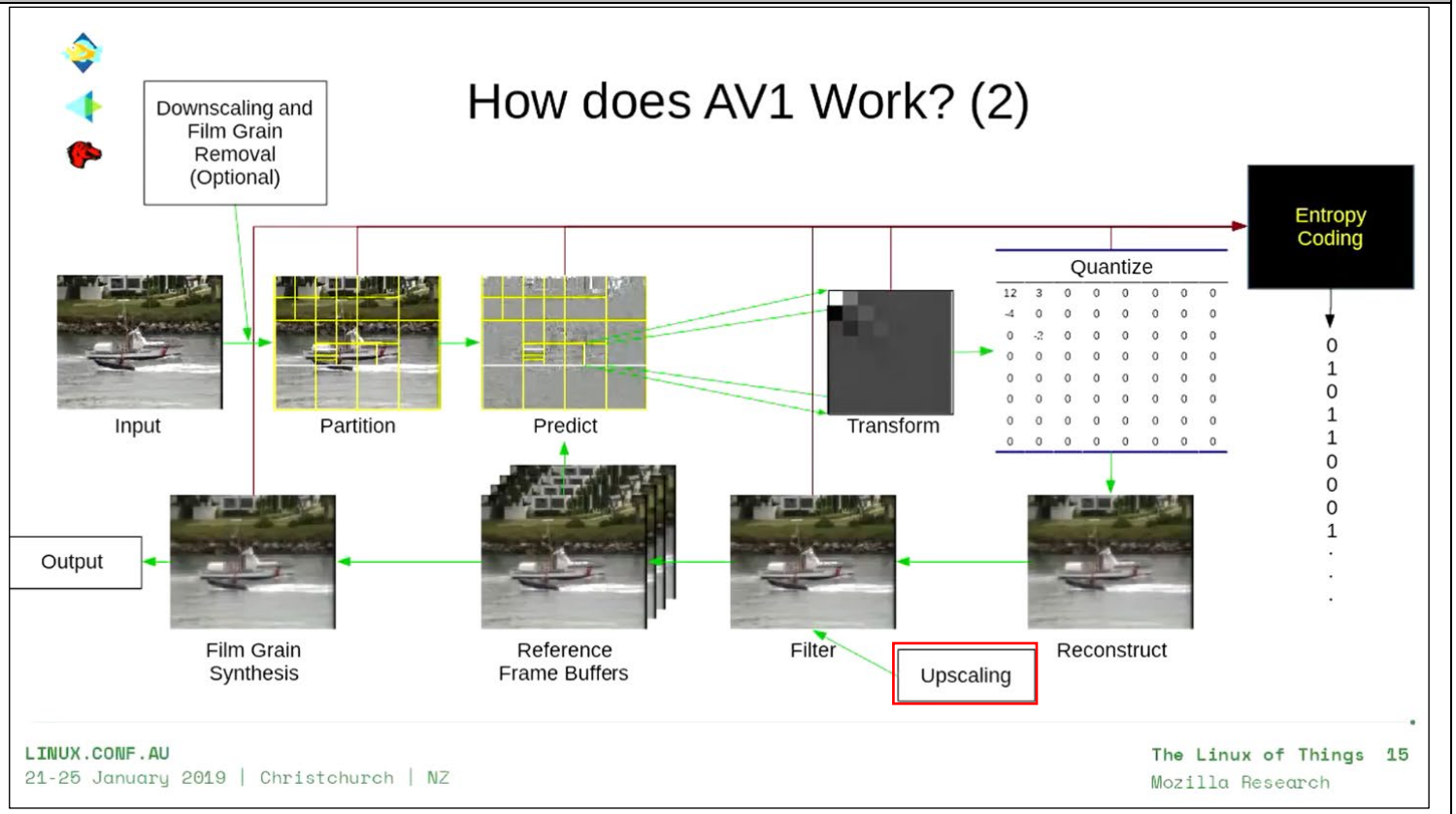
The array LrFrame is set equal to a copy of UpscaledCdefFrame. (The contents of LrFrame will later be overwritten for blocks that require restoration filtering.)

If UsesLr is equal to 0, then the process returns immediately after performing this copy.

Otherwise, loop restoration is applied as follows:

```
for ( y = 0; y < FrameHeight; y += MI_SIZE ) {
  for ( x = 0; x < UpscaledWidth; x += MI_SIZE ) {
    for ( plane = 0; plane < NumPlanes; plane++ ) {
      if ( FrameRestorationType[ plane ] != RESTORE_NONE ) {
        row = y >> MI_SIZE_LOG2
        col = x >> MI_SIZE_LOG2
        loop_restore_block( plane, row, col )
      }
    }
  }
}
```

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, an interpolation of pixels is applied on the decoded intraframe pictures to create a super resolution enlarged decoded picture. In AV1, the interpolation process is performed by wiener filter process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

7.17.1. Loop restore block process

The inputs to this process are:

- a variable plane specifying whether the process is filtering Y, U, or V samples,
- variables row and col specifying the location of the block in units of 4x4 blocks in the upscaled luma plane.

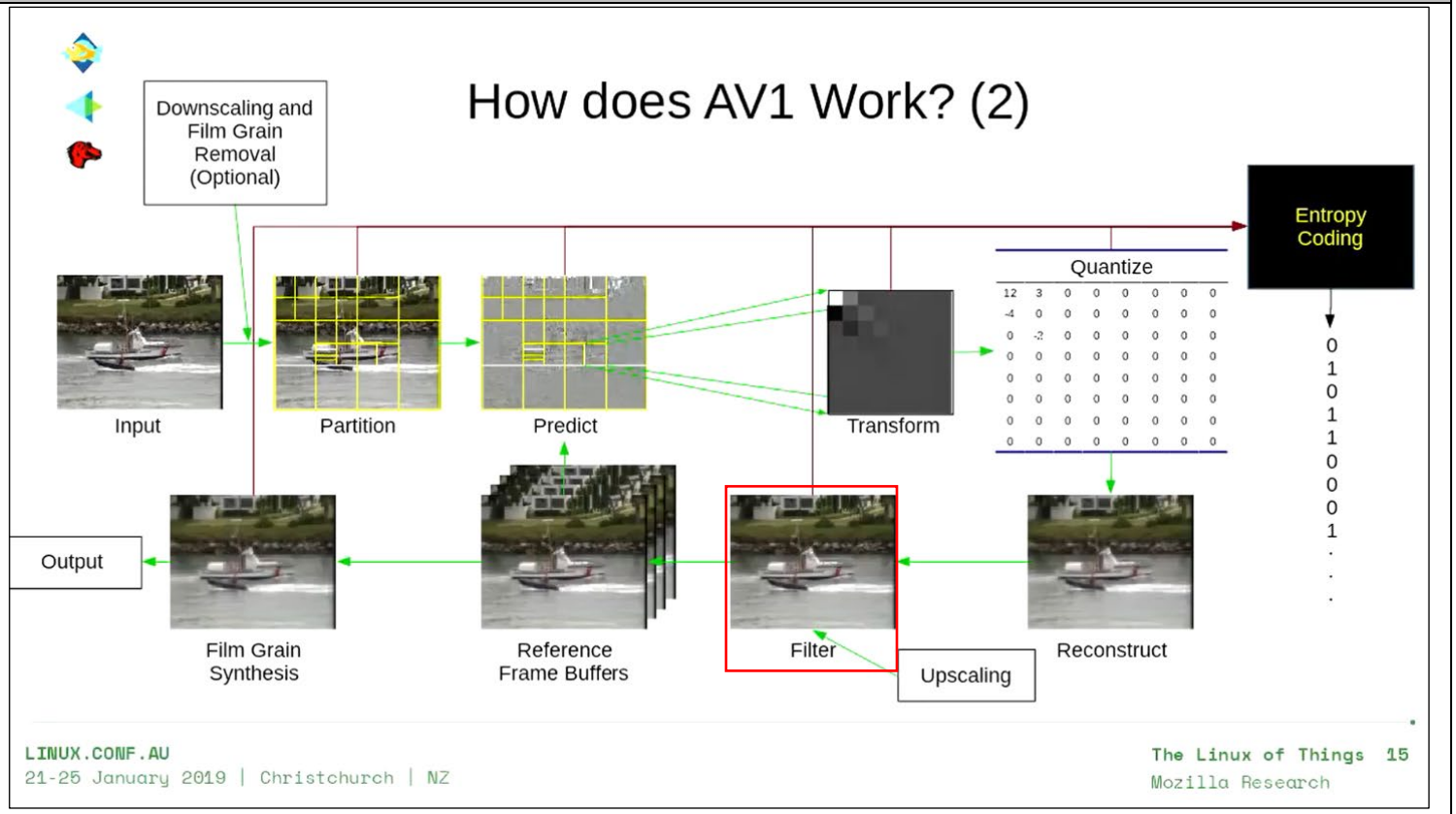
The output of this process are samples in LrFrame[plane].

The filter to used depends on rType as follows:

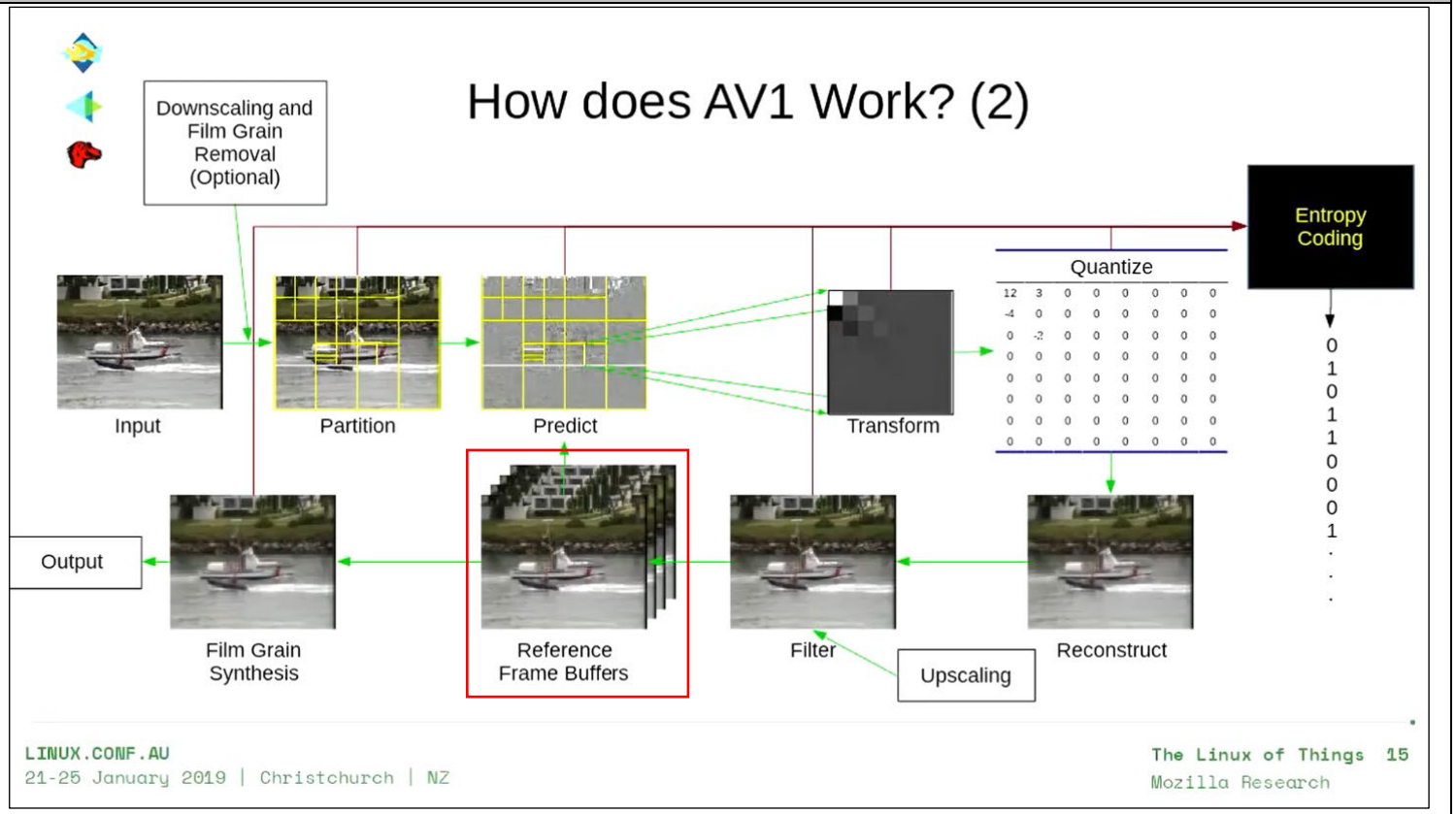
- If rType is equal to RESTORE_WIENER, the Wiener filter process specified in section 7.17.4 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise, if rType is equal to RESTORE_SGRPROJ, the self guided filter process specified in section 7.17.2 is invoked with plane, unitRow, unitCol, x, y, w, and h as inputs.
- Otherwise (rType is equal to RESTORE_NONE), no filtering is applied.

https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	7.17.4. Wiener filter process The inputs to this block are: • a variable plane specifying whether the process is filtering Y, U, or V samples, • variables unitRow and unitCol specifying the position of the loop restoration unit, • variables x and y specifying the position of the block in samples relative to the top-left corner of the current plane, • variables w and h specifying the size of the block in samples. The output from this process are modified samples in LrFrame. The sub-sample interpolation is effected via two one-dimensional convolutions. First a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction. https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
[4D] a step of acquiring the sequence of super-resolution enlarged decoded pictures created through the process for the prescribed super-resolution enlargement working on the sequence of super-resolution enlarged decoded pictures to implement a process for a first resolution conversion, creating a sequence of super-resolution decoded pictures with a	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of acquiring the sequence of super-resolution enlarged decoded pictures created through the process for the prescribed super-resolution enlargement working on the sequence of super-resolution enlarged decoded pictures to implement a process for a first resolution conversion, creating a sequence of super-resolution decoded pictures with a standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A acquires the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger. After the loop restoration process, the reconstructed intraframes are added to the reference buffer list which are further used for decoding interframes. The references pictures at the decoding side are scaled according to the resolution of the current interframe to be decoded. Since the first super-resolution enlarger provides an upscaled decoded reference picture, the reference pictures are downscaled to match the current interframe's resolution to be used as a reference picture for those interframes.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
standard resolution;	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

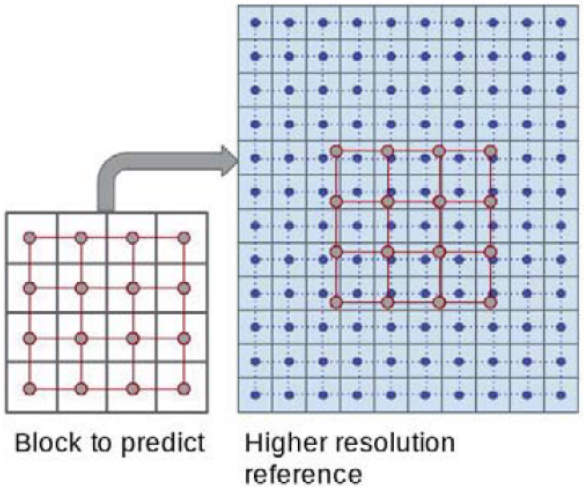
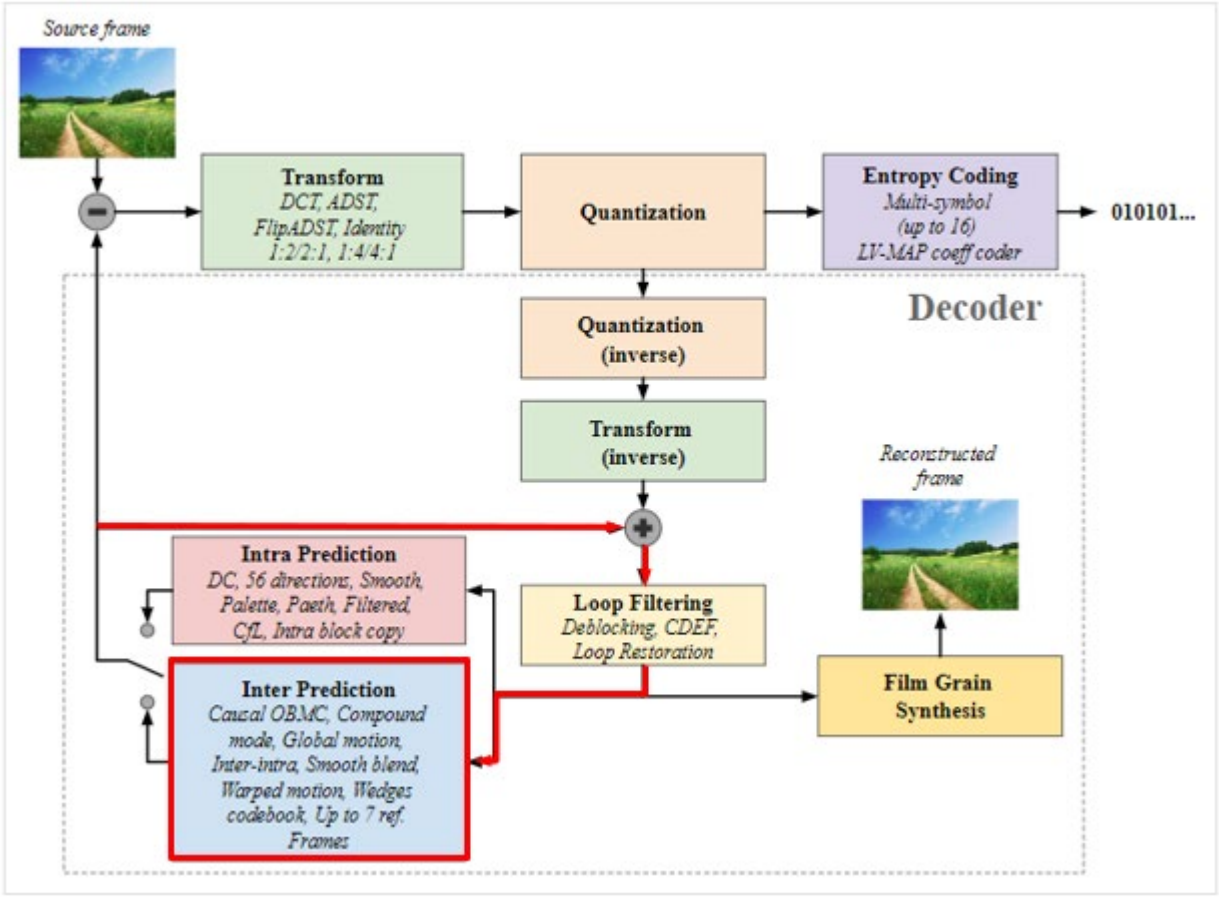


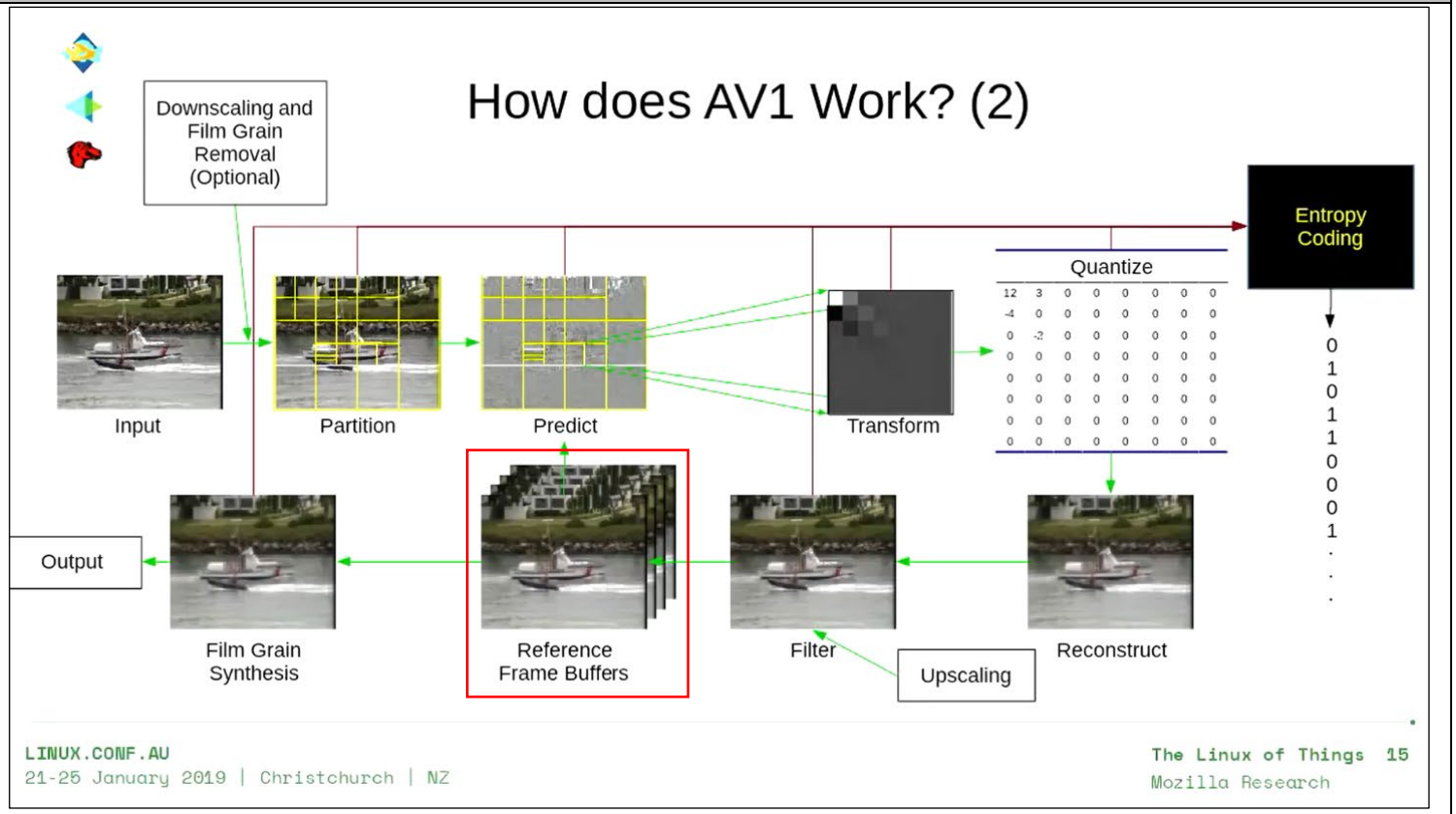
Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4E] a step of acquiring the second sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing as a set of decoding targets, the sequence of decoded pictures created with the standard resolution through the process for the	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of acquiring the second sequence of encoded bits obtained with a standard resolution through the process for the prescribed demultiplexing as a set of decoding targets, the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution through the process for the prescribed resolution conversion as a set of second reference pictures. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A acquires the second sequence of encoded bits obtained with a standard resolution at the demultiplexer (*e.g.*, sequence of interframes) as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder (*e.g.*, a sequence of decoded intraframes) as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures (*e.g.*, intraframes that are downscaled to match the current interframe's resolution to be used as a reference picture). As described above with respect to Claim Element [2A], two sequences of encoded bits are outputted, where the second sequence of encoded bits is the sequence of interframe bits. This sequence of interframe bits is a set of decoding targets because reconstructed interframes without super-resolution active (*i.e.*, the second sequence of encoded bits obtained with a standard resolution) are used for inter-prediction of the current frame in the decoding process.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
prescribed first decoding as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution through the process for the prescribed resolution conversion as a set of second reference pictures, and	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ Further, as described above with respect to Claim Element [2B], a sequence of decoded pictures (based on the first sequence of intraframe bits) is created with the standard resolution at the first decoder as a set of first reference pictures since these intraframes create the reference frames for the inter prediction described above.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	6.8.7. Superres params semantics **use_superres** equal to 0 indicates that no upscaling is needed. use_superres equal to 1 indicates that upscaling is needed. **coded_denom** is used to compute the amount of upscaling. **SuperresDenom** is the denominator of a fraction that specifies the ratio between the superblock width before and after upscaling. The numerator of this fraction is equal to the constant SUPERRES_NUM.[7] https://aomediacodec.github.io/av1-spec/av1-spec.pdf Moreover, as described above with respect to Claim Elements [4C] and [4D], the sequence of super-resolution decoded pictures with the standard resolution is converted. The AV1 decoder pipeline, with super-resolution active, produces decoded frames which are references that are super-resolved and then downscaled to match the current frame resolution (*i.e.*, the set of super-resolution enlarged and converted decoded pictures from the second resolution converter that acts as a set of second reference pictures).

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	A. Scaled Prediction In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

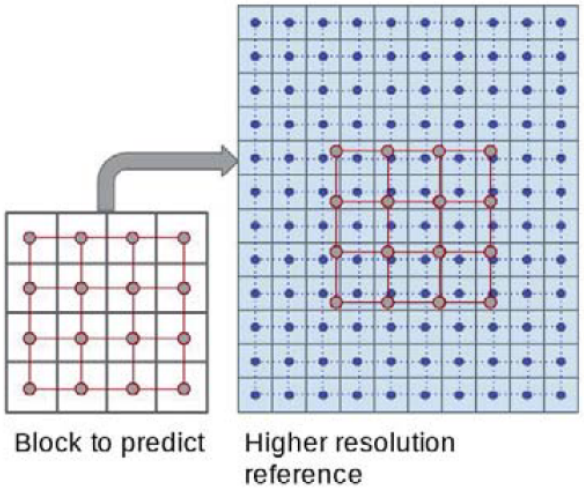
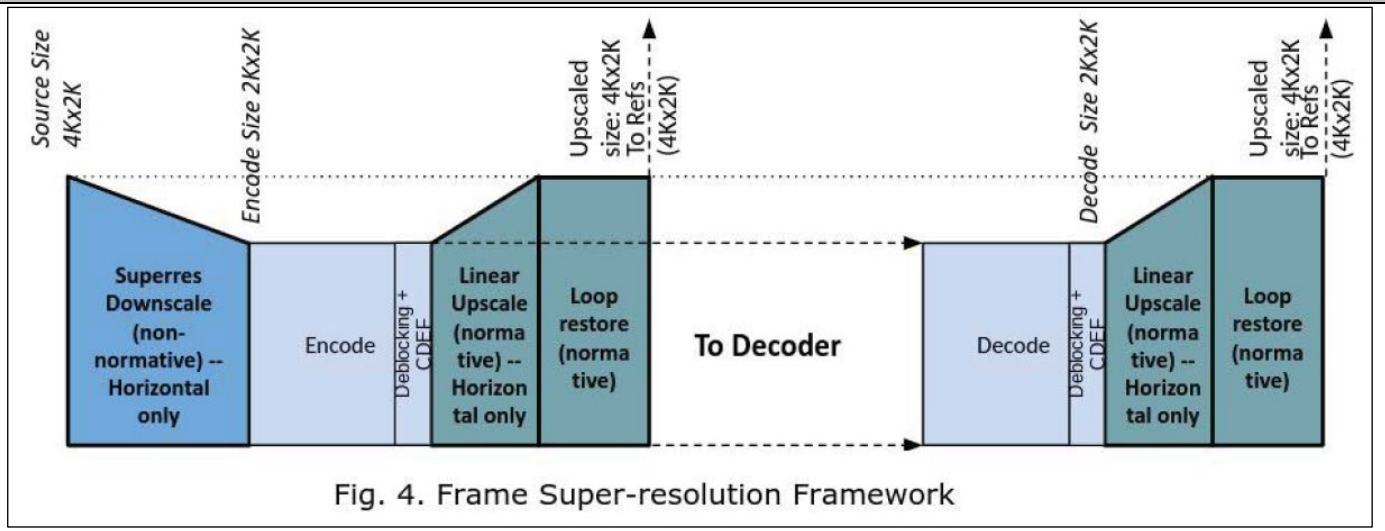


Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Therefore, both the sequence of decoded pictures created initially at standard resolution and the sequence of decoded pictures created by first upscaling and then downscaling, are sets of reference pictures acquired by the second decoder along with the second sequence comprising interframe pictures to perform the requisite interframe prediction. *B.1 Framework At a Glance* Fig. 4 depicts the overall in-loop filtering pipeline in AV1 and the super-resolution framework. According to this framework, on the encoder side, a source frame may be first downscaled non-normatively and encoded at a lower resolution. The deblocking filter, and another filter called Constrained Directional Enhancement Filter (CDEF) [17] supported in AV1, are then used to remove blocking and ringing artifacts while preserving edges, at the lower resolution. This is followed by a normative Linear Upsampling

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	process to full source resolution. Finally, another AV1 tool called Loop Restoration (LR) [18] is applied at full resolution to recover the high frequencies lost during the prior downscaling and quantization processes. The loop restored frame updates the reference buffer store for encoding subsequent frames. A similar process is followed on the decoder side as well. Note that the Deblocking, CDEF, Linear Upscaling and LR in order are all part of the normative in-loop filtering process in AV1, where each filter can be turned on or off per frame. https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Fig. 4. Frame Super-resolution Framework https://ieeexplore.ieee.org/document/8954553 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4F] selecting one of the set of first reference pictures and the set of second reference pictures based on reference	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, based on the set of decoding targets and the set of selected reference pictures to create a sequence of super-resolution pictures decoded with the standard resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, based on the set of decoding targets and the set of selected reference pictures to create a sequence of super-resolution pictures decoded with the standard resolution; and	This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A selects one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information (*e.g.*, selected based on a reference index) to implement a combination of processes for a prescribed prediction (*e.g.*, reconstructed frames without super-resolution are used for inter prediction of the current frame) and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution (*e.g.*, downscaled) based on the set of decoding targets (*e.g.*, resolution of the current frame) and the set of selected reference pictures (*e.g.*, the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index). As shown below, the AV1 codec supports the ability to predict pictures across scales using upscalers and downscalers (*i.e.*, a reference picture at a first resolution can be used to predict a picture at a second resolution). The second decoder waits for the current interframe to be decoded as received from the demultiplexer. When it receives the Interframe, the frame can be decoded based on the relevant reference Intraframe(s) - super-resolved or non-super-resolved, selected based on reference index.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

A. Scaled Prediction

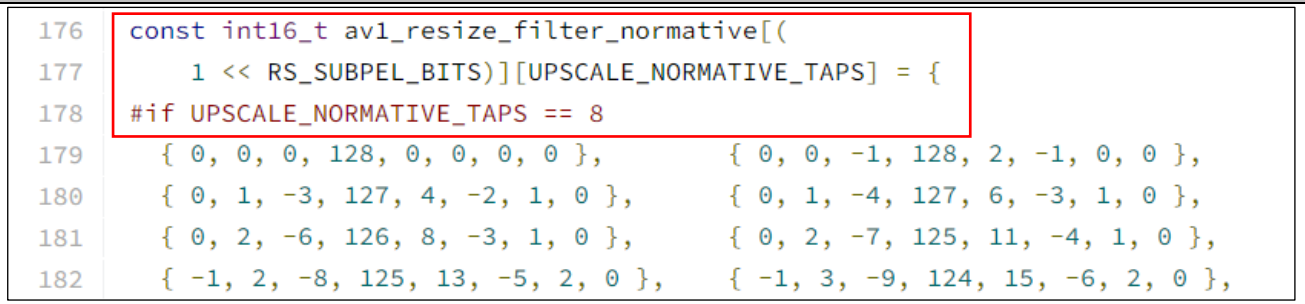
In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

https://ieeexplore.ieee.org/document/8954553

Since AV1 specification allow each frame to either be normally decoded or decoded with super-resolve steps, it is implied that the reference picture buffer set consists of both non-super-resolved and super-resolved intraframes that can be used as reference pictures. For the second decoder to decode the current interframe, the reference frame is selected based on the reference index. Thus, the reference index that indicates which reconstructed frame is selected for prediction is implied as 'the reference picture selection information' that is sent in the encoded bitstream.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	ref_frame_idx[i] specifies which reference frames are used by inter frames. It is a requirement of bitstream conformance that RefValid[ref_frame_idx[i]] is equal to 1, and that the selected reference frames match the current frame in bit depth, profile, chroma subsampling, and color space. **Note:** Syntax elements indicate a reference (such as LAST_FRAME, ALTREF_FRAME). These references are looked up in the ref_frame_idx array to find which reference frame should be used during inter prediction. There is no requirement that the values in ref_frame_idx should be distinct. https://aomediacodec.github.io/av1-spec/av1-spec.pdf Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4G] a step of acquiring the sequence of decoded pictures created with the standard resolution through the process for the prescribed first decoding, working on the	The Samsung Accused Products include a recording medium storing a moving picture decoding program comprising a non-transitory computer-readable medium configured to have a computer execute a step of acquiring the sequence of decoded pictures with the standard resolution from the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution. Samsung directly infringes and induces infringement by using, importing, testing, selling, and/or offering for sale devices which have the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A acquires the sequence of decoded pictures with the standard resolution from the first decoder (*e.g.*, a sequence of decoded intraframes) to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
sequence of decoded pictures to implement an interpolation of pixels with the second enlargement creating a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution,	decoded pictures with a high resolution as a second resolution higher than the standard resolution (*e.g.*, a sequence of upscaled decoded intraframes). Different 8-tap filter coefficients can be used for upscaling of the decoded frame. Google Git aomedia / aom / refs/heads/main / . / av1 / common / **resize.c** blob: f4bfcd062af2a6c4643314ec1501aa9c27306aa2 [file] [log] [blame] 1 /* 2 * Copyright (c) 2016, Alliance for Open Media. All rights reserved 3 * 4 * This source code is subject to the terms of the BSD 2 Clause License and 5 * the Alliance for Open Media Patent License 1.0. If the BSD 2 Clause License 6 * was not distributed with this source code in the LICENSE file, you can 7 * obtain it at www.aomedia.org/license/software. If the Alliance for Open 8 * Media Patent License 1.0 was not distributed with this source code in the 9 * PATENTS file, you can obtain it at www.aomedia.org/license/patent. 10 */

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://aomedia.googlesource.com/aom/+/refs/heads/main/av1/common/resize.c The AV1 codec accomplishes prediction across frames of different resolutions through its normative upscaling filters. For example, intraframes or Alternate Reference Frames, which are normally decoded by the first decoder, may require upscaling to match resolution of current frame. Since the AV1 codec allows switching between resolutions and using references of different resolutions, and coding can be done in an arbitrary resolution, a reconstructed frame which is to be used as reference might be present in a reconstructed-reference buffer whose resolution is different (lower) than the resolution of the interframe currently being decoded, for which the intraframe is an indexed reference. In such a scenario, a resolution conversion (*e.g.*, an upscale) would be required for decoding the current interframe. Such a resolution upscale is the AV1 codec's second resolution converter.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

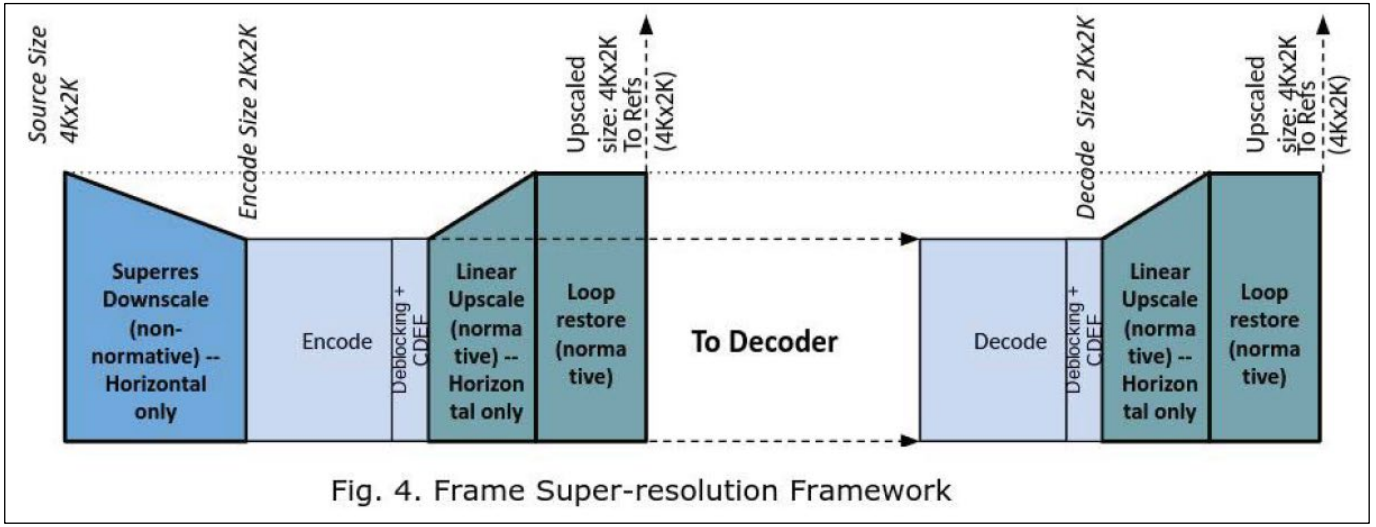
A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

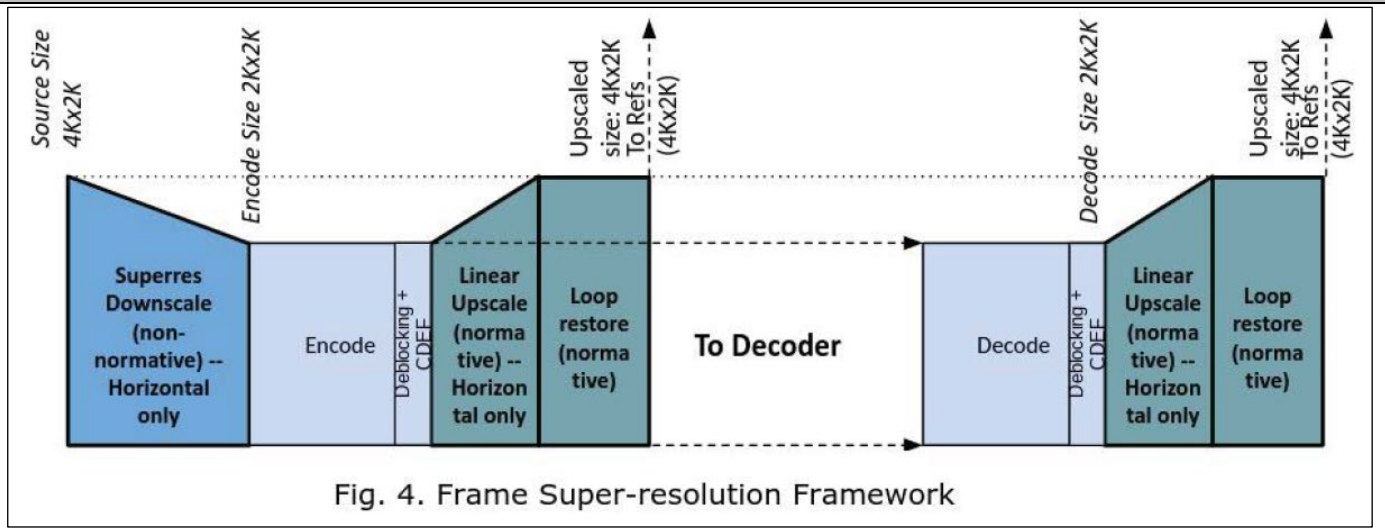
https://ieeexplore.ieee.org/document/8954553

Note that the AV1 bit-stream is flexible enough that each coded frame can use (a) horizontal-only super-resolution mode as described above, (b) code a frame at an arbitrary resolution within the ratio constraints, (c) a combination of both, or (d) no scaling at all. An intelligent encoder can utilize these options as additional knobs to be

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4H] wherein the first resolution is different from the second resolution,	The first resolution is different from the second resolution in the Samsung Accused Products. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, and as described by element [4G], the first resolution in the AV1 codec used by the Samsung 8K Neo QLED QN800A is different from the second resolution. Specifically, the second resolution is a "high resolution" that is "higher than the standard resolution," which is the first resolution. The first resolution is "2K x 2K" while the higher second resolution is "4K x 2K," as shown below.  Fig. 4. Frame Super-resolution Framework

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	https://ieeexplore.ieee.org/document/8954553 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4l] wherein the standard resolution of the second sequence of encoded bits is the same as the standard resolution of the super-resolution decoded pictures,	The standard resolution of the second sequence of encoded bits is the same as the standard resolution of the super-resolution decoded pictures in the Samsung Accused Products. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. As shown below, the standard resolution of the second sequence of encoded bits (*i.e.*, "Encode Size 2K x 2K") is the same as the standard resolution of the super-resolution decoded pictures (*i.e.*, "Decode Size 2K x 2K").

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 Fig. 4. Frame Super-resolution Framework https://ieeexplore.ieee.org/document/8954553 Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
[4J] and wherein the set of decoding targets, the set of first reference pictures, and the set of second reference	The set of decoding target pictures, the set of first reference pictures, and the set of second reference pictures in the Samsung Accused Products have the same value in spatial resolution. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. For example, the Samsung 8K Neo QLED QN800A upscales or downscales the reference pictures to match the resolution of the decoding targets. That is, the AV1 codec in the Samsung 8K Neo QLED QN800A scales (1) the set

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
pictures have the same value in spatial resolution.	of decoding target pictures; (2) the set of first reference pictures, and (3) the set of second reference pictures to have the same value in spatial resolution. As described below, "the motion vectors transmitted in the bitstream are always at the source resolution."

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

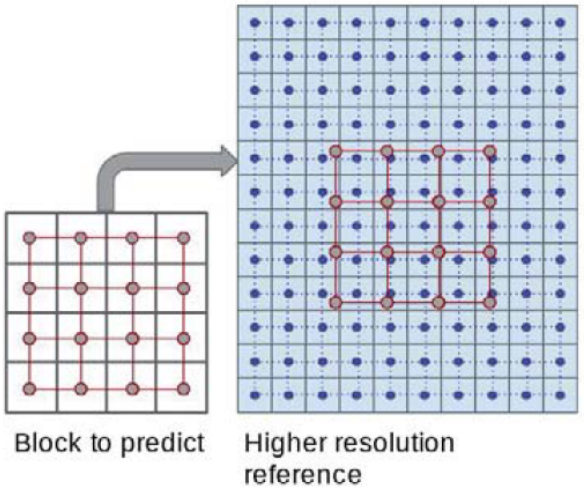
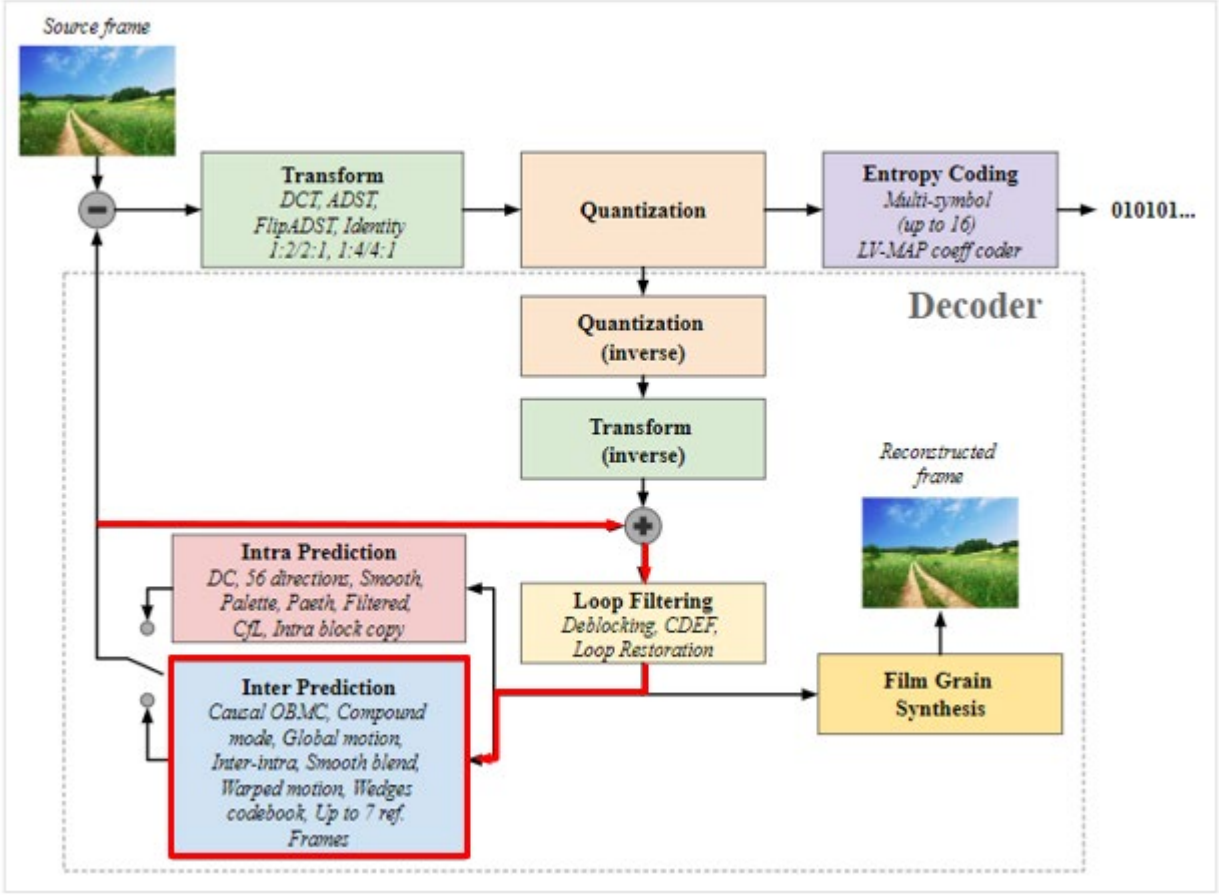


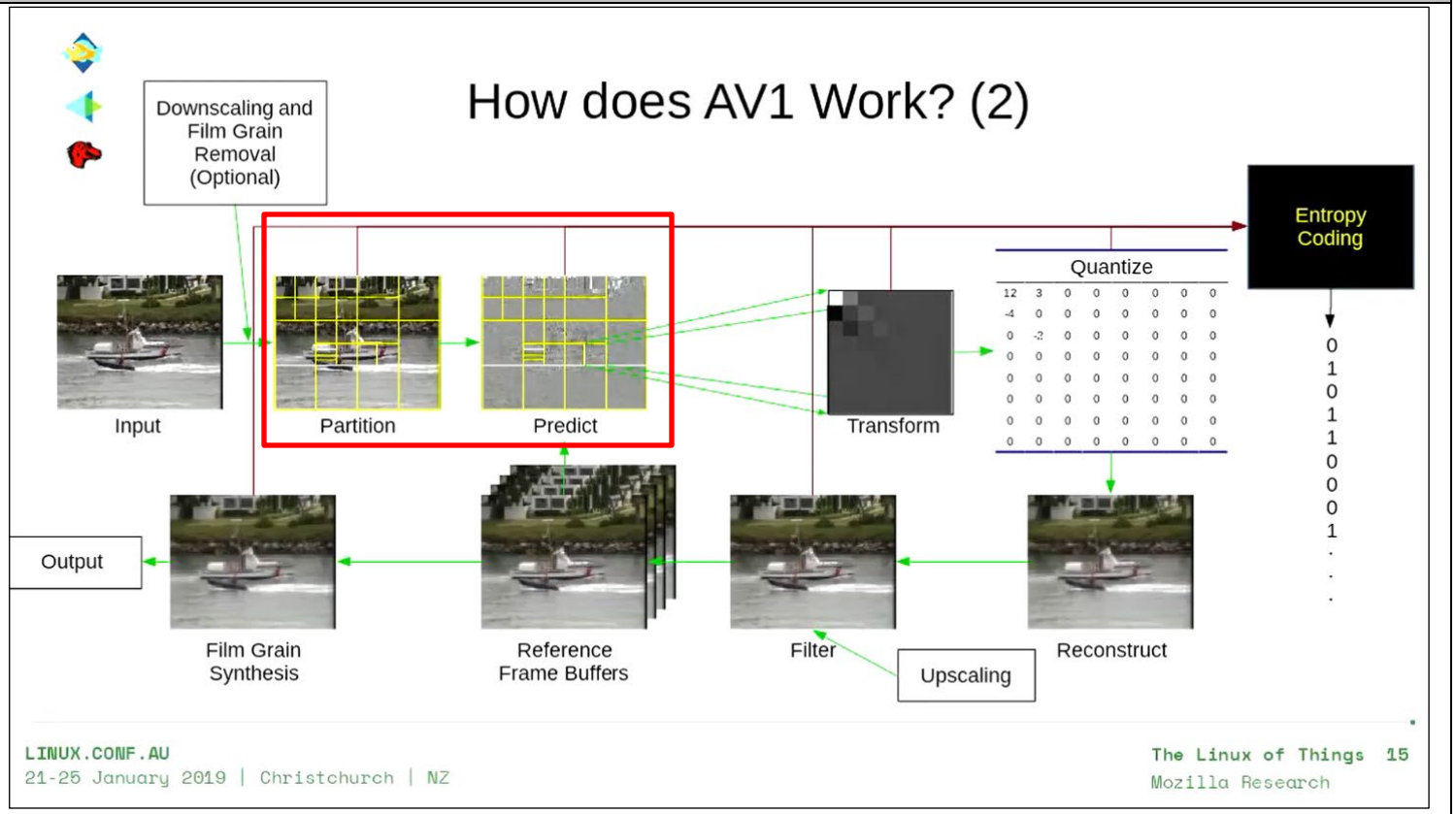
Fig. 2. Predicting a block from different (higher shown) resolution reference

https://ieeexplore.ieee.org/document/8954553

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	Further, to the extent this element is performed at least in part by software source code, ACT reserves the right to supplement these contentions pursuant to production of such source code and to the extent Defendants require additional information in accordance with P.R. 3-1 and for any other reasons.
11. The moving picture decoding system according to claim 2, wherein the second decoder divides a decoding target into a plurality of regions to select one of the set of first reference pictures and the set of second reference pictures for each divided region.	The Samsung Accused Products comprise a second decoder which divides a decoding target into a plurality of regions to select one of the set of first reference pictures and the set of second reference pictures for each divided region. Samsung directly infringes by using, importing, testing, selling, and/or offering for sale devices having the AV1 codec. This element is infringed literally, or in the alternative, under the doctrine of equivalents. A second decoder is configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer as a set of decoding targets for the purpose of interframe prediction using a set of reference pictures.

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/ The second decoder is configured to select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information. One of the reference picture selection information is a divided region of the decoding target. As part of the inter prediction process, several variables

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	specifying the region to be predicted are noted. Among these are "variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted" and "variables w and h specifying the width and height of the region to be predicted." 7.11.3. Inter prediction process 7.11.3.1. General The inter prediction process is invoked for inter coded blocks and interintra blocks. The inputs to this process are: • a variable plane specifying which plane is being predicted, • variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the region to be predicted, • variables w and h specifying the width and height of the region to be predicted, • variables candRow and candCol specifying the location (in units of 4x4 blocks) of the motion vector information to be used. The outputs of this process are predicted samples in the current frame CurrFrame. This process is triggered by a function call to predict_inter. https://aomediacodec.github.io/av1-spec/av1-spec.pdf

Claim – 10,218,995	Exemplary Supporting Evidence Regarding Samsung's Accused Products
	 https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/