Integrated Circuits and Systems

Vivienne Sze
Madhukar Budagavi
Gary J. Sullivan *Editors*

High Efficiency Video Coding (HEVC)

Algorithms and Architectures

Springer

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00001

Integrated Circuits and Systems

Series Editor
Anantha P. Chandrakasan
Massachusetts Institute of Technology
Cambridge, Massachusetts

For further volumes:
http://www.springer.com/series/7236

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00002

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00003

Vivienne Sze • Madhukar Budagavi
Gary J. Sullivan
Editors

High Efficiency Video Coding (HEVC)

Algorithms and Architectures

Springer

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00004

Editors
Vivienne Sze
Department of Electrical Engineering
and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

Madhukar Budagavi
Texas Instruments Inc.
Dallas, TX, USA

Gary J. Sullivan
Microsoft Corp.
Redmond, WA, USA

ISSN 1558-9412
ISBN 978-3-319-06894-7 ISBN 978-3-319-06895-4 (eBook)
DOI 10.1007/978-3-319-06895-4
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014930758

© Springer International Publishing Switzerland 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00005

Preface

Advances in video compression, which have enabled us to squeeze more pixels through bandwidth-limited channels, have been critical in the rapid growth of video usage. As we continue to push for higher coding efficiency, higher resolution and more sophisticated multimedia applications, the required number of computations per pixel and the pixel processing rate will grow exponentially. The High Efficiency Video Coding (HEVC) standard, which was completed in January 2013, was developed to address these challenges. In addition to delivering improved coding efficiency relative to the previous video coding standards, such as H.264/AVC, implementation-friendly features were incorporated into the HEVC standard to address the power and throughput requirements for many of today's and tomorrow's video applications.

This book is intended for readers who are generally familiar with video coding concepts and are interested in learning about the features in HEVC (especially in comparison to H.264/MPEG-4 AVC). It is meant to serve as a companion to the formal text specification and reference software. In addition to providing a detailed explanation of the standard, this book also gives insight into the development of various tools, and the trade-offs that were considered during the design process. Accordingly, many of the contributing authors are leading experts who were directly and deeply involved in the development of the standard itself.

As both algorithms and architectures were considered in the development of the HEVC, this aims to provide insight from both fronts. The first nine chapters of the book focus on the algorithms for the various tools in HEVC, and the techniques that were used to achieve its improved coding efficiency. The last two chapters address the HEVC tools from an architectural perspective and discuss the implementation considerations for building hardware to support HEVC encoding and decoding.

In addition to reviews from contributing authors, we would also like to thank the various external reviewers for their valuable feedback, which has helped improve the clarity and technical accuracy of the book. These reviewers include Yu-Hsin Chen,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00006

Chih-Chi Cheng, Keiichi Chono, Luis Fernandez, Daniel Finchelstein, Hun-Seok Kim, Hyungjoon Kim, Yasutomo Matsuba, Akira Osamoto, Rahul Rithe, Mahmut Sinangil, Hideo Tamama, Ye-Kui Wang and Minhua Zhou.

Cambridge, MA, USA — Vivienne Sze
Dallas, TX, USA — Madhukar Budagavi
Redmond, WA, USA — Gary J. Sullivan

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00007

About the Editors

Vivienne Sze is an Assistant Professor at the Massachusetts Institute of Technology (MIT) in the Electrical Engineering and Computer Science Department. Her research interests include energy-aware signal processing algorithms, and low-power circuit and system design for portable multimedia applications. Prior to joining MIT, she was with the R&D Center at Texas Instruments (TI), where she represented TI in the JCT-VC committee of ITU-T and ISO/IEC standards body during the development of HEVC (ITU-T H.265 | ISO/IEC 23008-2). Within the committee, she was the Primary Coordinator of the core experiments on coefficient scanning and coding and Chairman of ad hoc groups on topics related to entropy coding and parallel processing. Dr. Sze received the Ph.D. degree in Electrical Engineering from MIT. She has contributed over 70 technical documents to HEVC, and has published over 25 journal and conference papers. She was a recipient of the 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2008 A-SSCC Outstanding Design Award. In 2011, she received the Jin-Au Kong Outstanding Doctoral Thesis Prize in Electrical Engineering at MIT for her thesis on "Parallel Algorithms and Architectures for Low Power Video Decoding".

Madhukar Budagavi is a Senior Member of the Technical Staff at Texas Instruments (TI) and leads Compression R&D activities in the Embedded Processing R&D Center in Dallas, TX, USA. His responsibilities at TI include research and development of compression algorithms, embedded software implementation and prototyping, and video codec SoC architecture for TI products in addition to video coding standards participation. Dr. Budagavi represents TI in ITU-T and ISO/IEC international video coding standardization activity. He has been an active participant in the standardization of HEVC (ITU-T H.265 | ISO/IEC 23008-2) next-generation video coding standard by the JCT-VC committee of ITU-T and ISO/IEC. Within the JCT-VC committee he has helped coordinate sub-group activities on spatial transforms, quantization, entropy coding, in-loop filtering, intra prediction, screen content coding and scalable HEVC (SHVC). Dr. Budagavi received the Ph.D. degree in Electrical Engineering from Texas A&M University. He has published 6 book chapters and over 35 journal and conference papers. He is a Senior Member of the IEEE.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00008

Gary J. Sullivan is a Video and Image Technology Architect at Microsoft Corporation in its Corporate Standardization Group. He has been a longstanding chairman or co-chairman of various video and image coding standardization activities in ITU-T VCEG, ISO/IEC MPEG, ISO/IEC JPEG, and in their joint collaborative teams since 1996. He is best known for leading the development of the AVC (ITU-T H.264 | ISO/IEC 14496-10) and HEVC (ITU-T H.265 | ISO/IEC 23008-2) standards, and the extensions of those standards for format application range enhancement, scalable video coding, and 3D/stereoscopic/multiview video coding. At Microsoft, he has been the originator and lead designer of the DirectX Video Acceleration (DXVA) video decoding feature of the Microsoft Windows operating system. Dr. Sullivan received the Ph.D. degree in Electrical Engineering from the University of California, Los Angeles. He has published approximately 10 book chapters and prefaces and 50 conference and journal papers. He has received the IEEE Masaru Ibuka Consumer Electronics Technical Field Award, the IEEE Consumer Electronics Engineering Excellence Award, the Best Paper award of the IEEE *Trans. CSVT*, the INCITS Technical Excellence Award, the IMTC Leadership Award, and the University of Louisville J. B. Speed Professional Award in Engineering. The team efforts that he has led have been recognized by an ATAS Primetime Emmy Engineering Award and a pair of NATAS Technology & Engineering Emmy Awards. He is a Fellow of the IEEE and SPIE.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00009

Contents

1 Introduction 1
Gary J. Sullivan

2 HEVC High-Level Syntax 13
Rickard Sjöberg and Jill Boyce

3 Block Structures and Parallelism Features in HEVC 49
Heiko Schwarz, Thomas Schierl, and Detlev Marpe

4 Intra-Picture Prediction in HEVC 91
Jani Lainema and Woo-Jin Han

5 Inter-Picture Prediction in HEVC 113
Benjamin Bross, Philipp Helle, Haricharan Lakshman, and Kemal Ugur

6 HEVC Transform and Quantization 141
Madhukar Budagavi, Arild Fuldseth, and Gisle Bjøntegaard

7 In-Loop Filters in HEVC 171
Andrey Norkin, Chih-Ming Fu, Yu-Wen Huang, and Shawmin Lei

8 Entropy Coding in HEVC 209
Vivienne Sze and Detlev Marpe

9 Compression Performance Analysis in HEVC 275
Ali Tabatabai, Teruhiko Suzuki, Philippe Hanhart, Pavel Korshunov, Touradj Ebrahimi, Michael Horowitz, Faouzi Kossentini, and Hassene Tmar

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00010

10 Decoder Hardware Architecture for HEVC 303
Mehul Tikekar, Chao-Tsung Huang, Chiraag Juvekar, Vivienne Sze, and Anantha Chandrakasan

11 Encoder Hardware Architecture for HEVC 343
Sung-Fang Tsai, Cheng-Han Tsai, and Liang-Gee Chen

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00011

Chapter 1
Introduction

Gary J. Sullivan

Abstract The new HEVC standard enables a major advance in compression relative to its predecessors, and its development was a large collaborative effort that distilled the collective knowledge of the whole industry and academic community into a single coherent and extensible design. This book collects the knowledge of some of the key people who have been directly involved in developing or deploying the standard to help the community understand the standard itself and its implications. A detailed presentation is provided for each of the standard's fundamental building blocks and how they fit together to make HEVC the powerful package that it is. The compression performance of the standard is analyzed, and architectures for its implementation are described. We believe this book provides important information for the community to help ensure the broad success of HEVC as it emerges in a wide range of products and applications. The applications for HEVC will not only cover the space of the well-known current uses and capabilities of digital video—they will also include the deployment of new services and the delivery of enhanced video quality, such as the deployment of ultra-high-definition television (UHDTV) and video with higher dynamic range, a wider range of representable color, and greater representation precision than what is typically found today.

1.1 HEVC Background and Development

The standard now known as *High Efficiency Video Coding* (HEVC) [3] reflects the accumulated experience of about four decades of research and three decades of international standardization for digital video coding technology. Its development

G.J. Sullivan (✉)
Microsoft Corp., Redmond, WA, USA
e-mail: garysull@microsoft.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_1
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00012

was a massive undertaking that dwarfed prior projects in terms of the sheer quantity of engineering effort devoted to its design and standardization. The result is now formally standardized as ITU-T Recommendation H.265 and ISO/IEC International Standard 23008-2 (MPEG-H part 2). The first version of HEVC was completed in January 2013 (with final approval and formal publication following a few months later—specifically, ITU-T formal publication was in June, and ISO/IEC formal publication was in November). While some previous treatments of the HEVC standard have been published (e.g., [8]), this book provides a more comprehensive and unified collection of key information about the new standard that will help the community to understand it well and to make maximal use of its capabilities.

The HEVC project was formally launched in January 2010, when a joint Call for Proposals (CfP) [4, 6, 10] was issued by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). Before launching the formal CfP, both organizations had conducted investigative work to determine that it was feasible to create a new standard that would substantially advance the state of the art in compression capability—relative to the prior major standard known as H.264/MPEG-4 *Advanced Video Coding* (AVC) [2, 7, 9] (the first version of which was completed in May 2003).

One notable aspect of the investigative work toward HEVC was the "key technology area" (KTA) studies in VCEG that began around the end of 2004 and included the development of publicly-available KTA software codebase for testing various promising algorithm proposals. In MPEG, several workshops were held, and a Call for Evidence (CFE) was issued in 2009. When the two groups both reached the conclusion that substantial progress was possible and that working together on the topic was feasible, a formal partnership was established and the joint CfP was issued. The VCEG KTA software and the algorithmic techniques found therein were used as the basis of many of the proposals submitted in response to both the MPEG CfE and the joint CfP.

Interest in developing a new standard has been driven not only by the simple desire to improve compression as much as possible—e.g., to ease the burden of video on storage systems and global communication networks, but also to help enable the deployment of new services, including capabilities that have not previously been practical—such as ultra-high-definition television (UHDTV) and video with higher-dynamic range, wider color gamut, and greater representation precision than what is typically found today.

To formalize the partnership arrangement, a new joint organization was created, called the Joint Collaborative Team on Video Coding (JCT-VC). The JCT-VC met four times per year after its creation, and each meeting had hundreds of attending participants and involved the consideration of hundreds of contribution documents (all of which were made publicly available on the web as they were submitted for consideration).

The project had an unprecedented scale, with a peak participation reaching about 300 people and more than 1,000 documents at a single meeting. Meeting notes were publicly released on a daily basis during meetings, and the work continued between meetings, with active discussions by email on a reflector with a distribution list with

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00013

thousands of members, and with formal coordination between meetings in the form of work by "ad hoc groups" to address particular topics and "core experiments" to test various proposals. Essentially the entire community of relevant companies, universities, and other research institutions was attending and actively participating as the standard was developed.

There had been two previous occasions when the ITU's VCEG and ISO/IEC's MPEG groups had formed similar partnerships. One was AVC, about a decade earlier, and the other was what became known as MPEG-2 (which was Recommendation H.262 in the ITU naming convention), about a decade before that. Each of these had been major milestones in video coding history. About a decade before those was when the standardization of digital video began, with the creation of the ITU's Recommendation 601 in 1982 for uncompressed digital video representation and its Recommendation H.120 in 1984 as the first standard digital video compression technology—although it would not be until the second version of Recommendation H.261 was established in 1990 that a really adequate compression design would emerge (and in several ways, even the HEVC standard owes its basic design principles to the scheme found in H.261).

1.2 Compression Capability: The Fundamental Need

Uncompressed video signals generate a huge quantity of data, and video use has become more and more ubiquitous. There is also a constant hunger for higher quality video—e.g., in the form of higher resolutions, higher frame rates, and higher fidelity—as well as a hunger for greater access to video content. Moreover, the creation of video content has moved from the being the exclusive domain of professional studios toward individual authorship, real-time video chat, remote home surveillance, and even "always on" wearable cameras. As a result, video traffic is the biggest load on communication networks and data storage world-wide—a situation that is unlikely to fundamentally change; although anything that can help ease the burden is an important development. HEVC offers a major step forward in that regard [5].

Today, AVC is the dominant video coding technology used world-wide. As a rough estimate, about half the bits sent on communication networks world-wide are for coded video using AVC, and the percentage is still growing. However, the emerging use of HEVC is likely to be the inflection point that will soon cause that growth to cease as the next generation rises toward dominance.

MPEG-2 basically created the world of digital video television as we know it, so while AVC was being developed, some people doubted that it could achieve a similar degree of ubiquity when so much infrastructure had been built around the use of MPEG-2. Although it was acknowledged that AVC might have better compression capability, some thought that the entrenched universality of MPEG-2 might not allow a new non-compatible coding format to achieve "critical mass".

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00014

When completed, AVC had about twice the compression capability of MPEG-2—i.e., one could code video using only about half the bit rate while still achieving the same level of quality—so that one could send twice as many TV channels through a communication link or store twice as much video on a disc without sacrificing quality. Alternatively, the improved compression capability could be used to provide higher quality or enable the use of higher picture resolution or higher frame rates than would otherwise be possible. AVC also emerged at around the same time that service providers and disc storage format designers were considering a transition to offer higher resolution "HDTV" rather than their prior "standard definition" television services. Once system developers realized that they needed to store and send twice as much data if they were going to use MPEG-2 instead of AVC for whatever video service they were trying to provide, most of them decided they needed to find a transition path to AVC. While MPEG-2 video remains a major presence today for legacy compatibility reasons, it is clearly fading away in terms of importance.

HEVC offers the same basic value proposition today that AVC did when it emerged—i.e., a doubling of compression capability. It can compress video about twice as much as AVC without sacrificing quality, or it can alternatively be used to enable delivery of higher resolutions and frame rates—or other forms of higher quality, such as a higher dynamic range or higher precision for improved color quality. It also comes at another time when new video services are emerging—this time for UHDTV, higher dynamic range, and wider color gamut.

Compression capability—also known as "coding efficiency" or "compression efficiency"—is the most fundamental driving force behind the adoption of modern digital video compression technology, and HEVC is exceptionally strong in that area. It is this meaning from which the High Efficiency Video Coding standard derives its name. However, it is also important to remember that the standard only provides encoders with the *ability* to compress video efficiently—it does not guarantee any particular level of quality, since it does not govern whether or not encoders will take full advantage of the capability of the syntax design (or whether or not they will use that syntax for other purposes such as enhanced loss robustness).

1.3 Collaborative Development, Interoperability, and Flexibility

As noted earlier, the HEVC standard was developed in an open process with very broad participation. This helped to ensure that the design would apply generically across a very broad range of applications, and that it was well studied and flexible and would not contain quirky shortcomings that could have been prevented by greater scrutiny during the design process.

Moreover, much of what can distinguish a good formal "standard" from simply any particular well-performing technology product is the degree to which *interop-*

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00015

erability is enabled across a breadth of products made by different entities. The goal of the HEVC standard is not just to compress video well, but also to enable the design to be used in many different products and services across a very wide range of application environments. No assumption is made that encoders and decoders will all work the same way—in fact, a great deal of intentional flexibility is built into the design. Indeed, strictly speaking, it is incorrect to refer to a standard such as HEVC or AVC as a "codec"—since the standard does not specify an encoder and a decoder. Instead, it specifies only a common format—a common *language* by which encoding and decoding systems, each made separately using different computing architectures and with different design constraints and priorities, can nevertheless communicate effectively.

A great deal of what characterizes a product has been deliberately left outside the scope of the standard, particularly including the following:

- **The entire encoding process**: Encoder designers are allowed to encode video using any searching and decision criteria they choose—so long as the format of their output conforms to the format specifications of the standard. This particularly includes the relative prioritization of various bitstream characteristics—the standard allows encoders to be designed primarily for low complexity, primarily for high coding efficiency, primarily to enable good recovery from data losses, primarily to minimize real-time communication latency, etc.
- **Many aspects of the decoding process**: When presented with a complete and uncorrupted coded bitstream, the standard requires decoders to produce particular decoded picture data values at some processing stage as their theoretical "output"; however, it does not require the decoders to use the same exact processing steps in order to produce that data.
- **Data loss and corruption detection and recovery**: The standard does not govern what a decoder will do if it is presented with incomplete or corrupted video data. However, in real-world products, coping with imperfect input is a fundamental requirement.
- **Extra functionalities**: Operations such as random access and channel switching, "trick mode" operations like fast-forwarding and smooth rewind, and other functions such as bitstream splicing are all left out of the scope of the standard to allow products to use the coded data as they choose.
- **Pre-processing, post-processing, and display**: Deliberate alteration of encoder input data and post-decoding picture modification is allowed for whatever reason the designers may choose, and how the video is ultimately displayed (including key aspects such as the accuracy of color rendering) is each product's own responsibility.

All of this gives implementers a great deal of freedom and flexibility, while governing only what is absolutely necessary to establish the ability for data that is properly encoded by any "conforming" encoder to be decoded by any "conforming" decoder (subject to profile/tier/level compatibility as further discussed below). It does not necessarily make the job of the encoder and decoder designer especially easy, but it enables products made by many different people to communicate

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00016

effectively with each other. In some cases, some freedom that is provided in the video coding standard may be constrained in other ways, such as constraints imposed by other specifications that govern usage in particular application environments.

Another key element of a good international standard is the quality of its specification documentation and the availability of additional material to help implementers to use the design and to use it well. In the case of HEVC (and AVC and some other international standards before those), this includes the following:

- **The text specification itself**: in the case of HEVC version 1, the document [3] is about 300 pages of carefully-written (although dense and not necessarily easy to read) detailed specification text that very clearly describes all aspects of the standard.
- **Reference software source code**: a collaboratively developed software codebase that can provide a valuable example of how to use the standard format (for both encoding and decoding) and help clarify any ambiguities or difficulties of interpreting the specification document.
- **Conformance data test set**: a suite of tests to be performed to check implementations for proper conformance to the standard.
- **Other standards designed to work with the technology**: this includes many other industry specifications and formal standards that have been developed, maintained, and enhanced within the same broad industry community that developed the video coding specification itself—e.g., data multiplexing designs, systems signaling and negotiation mechanisms, storage formats, dynamic delivery protocols, etc.
- **Many supplemental publications in industry and academic research literature**: a diverse source of tutorial information, commentary, and exploration of the capabilities, uses, limitations, and possibilities for further enhancement of the design. This book, of course, is intended to become a key part of this phenomenon.

The syntax of the HEVC standard has been carefully designed to enable flexibility in how it is used. Thus, the syntax contains features that give it a unified syntax architecture that can be used in many different system environments and can provide customized tradeoffs between compression and other aspects such as robustness to data losses. Moreover, the high-level syntax framework of the standard is highly extensible and provides flexible mechanisms for conveying (standard or non-standard) supplemental enhancement information along with the coded video pictures.

Maintenance of the standard specifications (and the development of further enhancement extensions in a harmonized manner) is another significant part of the phenomenon of standardization best practices. In the case of HEVC, the standard, and the associated related standards, have been collaboratively developed by the most well-established committees in the area and with a commitment to follow through on the developments represented by the formal specifications.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00017

1.4 Complexity, Parallelism, Hardware, and Economies of Scale

When a technical design such as HEVC is new, its practicality for implementation is especially important. And when they emerged as new standards, H.261, MPEG-2, and AVC were each rather difficult to implement in decoders—they stretched the bounds of what was practical to produce at the time, although they each proved to be entirely feasible in short order. In each of those cases, major increases in computing power and memory capacity were needed to deploy the new technology. Of course, as time has moved forward, Moore's law has worked its magic, and what was once a major challenge has become a mundane expectation.

Thankfully, HEVC is less of a problem than its predecessors in that regard [1]. Although its decoding requirements do exceed those of the prior AVC standard, the increase is relatively moderate. The memory capacity requirement has not substantially increased beyond that for AVC, and the computational resource requirements for decoding are typically estimated in the range of 1.5–2 times those for AVC. With a decade of technology progress since AVC was developed, this makes HEVC decoding not really so much of a problem. The modesty of this complexity increase was the result of careful attention to practicality throughout the design process.

Moreover, the need to take advantage of parallel processing architectures was recognized throughout the development of HEVC, so it contains key new features—both large and small—that are friendly to parallel implementation. Each design element was inspected for potential serialized bottlenecks, which were avoided as much as possible. As parallelism is an increasingly important element of modern processing architectures, we are proud that its use has been deeply integrated into the HEVC design.

Another key issue is power consumption. Today's devices increasingly demand mobility and long battery life. It has already been well-demonstrated that HEVC is entirely practical to implement using only software—even for high-resolution video and even using only the computing resources found in typical laptops, tablets, and even mobile phones. However, the best battery life will be obtained by the use of custom silicon, and having the design stability, well-documented specification, and cross-product interoperability of a well-developed international standard will help convince silicon designers that investing in HEVC is appropriate. Once broad support in custom silicon is available from multiple vendor sources, economies of scale will further take hold and drive down the cost and power consumption to very low levels (aside, perhaps, for patent licensing costs, as further discussed below). Indeed, this is already evident, as some custom-silicon support is already emerging in products.

Encoding is more of a challenge than decoding—quite a substantial challenge, at this point. HEVC offers a myriad of choices to encoders, which must search among the various possibilities and decide which to use to represent their video most effectively. Although this is likely to present a challenge for some time to come,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00018

preliminary product implementations have already shown that HEVC encoding is entirely feasible. Moreover, experience has also shown that as time moves forward, the effectiveness of encoders to compress video within the constraints imposed by the syntax of a particular standard can continue to increase more and more, while maintaining compatibility with existing decoders. Indeed, encoders for MPEG-2 and AVC have continued to improve, despite the limitations of their syntax.

1.5 Profiles, Levels, Tiers, and Extensibility

Although we tend to think of a standard as a single recipe for guaranteed interoperability, some variation in capabilities is necessary to support a broad range of applications. In HEVC, as with some prior designs, this variation is handled by specifying multiple "profiles" and "levels". Moreover, for HEVC a new concept of "tiers" has been introduced. However, the diversity of separate potential "islands" of interoperability in version 1 of HEVC is quite modest—and depends on the intended applications in a straightforward manner. Only three profiles are found in the first version of the standard:

- **Main profile**: for use in the typical applications that are familiar to most consumers today. This profile represents video data with 8 bits per sample and the typical representation with a "luma" brightness signal and two "chroma" channels that have half the luma resolution both horizontally and vertically.
- **Main Still Picture profile**: for use as still photography for cameras, or for extraction of snapshots from video sequences. This profile is a subset of the capabilities of the Main profile.
- **Main 10 profile**: supporting up to 10 bits per sample of decoded picture precision. This profile provides increased bit depth for increased brightness dynamic range, extended color-gamut content, or simply higher fidelity color representations to avoid contouring artifacts and reduce rounding error. This profile is a superset of the capabilities of the Main profile.

However, the syntax design is highly extensible, and various other profiles are planned to be added in future extensions. The extensions under development include major efforts on extensions of the range of supported video formats (including higher bit depths and higher-fidelity chroma formats such as the use of full-resolution chroma), layered coding scalability, and 3D multiview video. The JCT-VC, and a new similar organization called the JCT-3V for 3D video work, have continued to meet at the same meeting frequency to develop these extensions and they remain very active in that effort—with more than 150 participants and more than 500 contribution documents per meeting.

While profiles define the syntax and coding features that can be used for the video content, a significant other consideration is the degree of capability within a given feature set. This is the purpose of "levels". Levels of capability are defined

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00019

to establish the picture resolution, frame rate, bit rate, buffering capacity, and other aspects that are matters of degree rather than basic feature sets. For HEVC, the lowest levels have only low resolution and low frame rate capability—e.g., a typical video format for level 1 may be only 176×144 resolution at 15 frames per second, whereas level 4.1 would be capable of 1920×1080 HDTV at 60 frames per second, and levels in version 1 are defined up to level 6.1, which is capable of 8192×4320 video resolution at up to 120 frames per second.

However, when defining the levels of HEVC, a problem was encountered between the demands of consumer use and those of professional use for similar picture resolutions and frame rates. In professional environments, much higher bit rates are needed for adequate quality than what would be necessary for consumer applications. The solution for this was to introduce the concept of "tiers". Several levels in HEVC have both a Main tier and a High tier of capability specified, based on the bit rates they are capable of handling.

1.6 Patent Rights Licensing

The consideration of a modern video coding design would be incomplete without some understanding of the costs of the patent rights needed to use it. Digital video technology is a subject of active research, investment, and innovation, and many patents have been filed on advances in this field.

The international standardization organizations have patent policies that require that technology cannot be included in a standard if the patent rights that are essential to its implementation are known to not be available for licensing to all interested parties on a world-wide basis under "reasonable and non-discriminatory" (RAND) licensing terms. The idea behind this is that anyone should be able to implement an international standard without being forced to agree to unreasonable business terms. In other respects, the major standardization bodies generally do not get involved in the licensing details for necessary patent rights—these are to be negotiated separately, between the parties involved, outside the standardization development process.

In recent history—e.g., for both AVC and MPEG-2, multiple companies have gotten together to offer "pooled" patent licensing as a "one-stop shop" for licensing the rights to a large number of necessary patents. A pool for HEVC patent licensing has also recently begun to be formed and has announced preliminary licensing terms. However, it is important for the community to understand that the formation of such a pool is entirely separate from the development of the standard itself. Patent holders are not required to join a pool, and even if they choose to join a pool, they may also offer licenses outside the pool as well—as such pools are non-exclusive licensing authorities. And licensees are thus not required to get their licensing rights through such a pool and can seek any rights that are required on a bilateral basis outside of the pool.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00020

Patent pools and standardization do not offer perfect answers to the sticky problems surrounding the establishment of known and reasonable costs for implementing modern digital video technology. In fact, a number of substantial disputes have arisen in relation to the previous major standards for video coding, and such disputes may occur for HEVC as well. However, proposed proprietary alternatives—including those asserted to be "open source" or "royalty free"—are not necessarily an improvement over that situation, as they bring with them their own legal ambiguity. For example, since those proprietary technologies are not generally standardized, such designs may carry no assurances of licensing availability or of licenses having "reasonable and non-discriminatory" terms.

It is likely to take some time for the industry to sort out the patent situation for HEVC, as has been the case for other designs. There is little clear alternative to that, since the only designs that are clearly likely to be free of patent rights are those that were developed so long ago that all the associated patents have expired—and such schemes generally may not have adequate technical capability. In regard to the previous major international standards, the industry has ultimately sorted out the business terms so that the technology could be widely used by all with reasonable costs and a manageable level of business risk—and we certainly hope that this will also be the case for HEVC.

1.7 Overview of This Book

This book collects together the key information about the design of the new HEVC standard, its capabilities, and its emerging use in deployed systems. It has been written by key experts on the subject—people who were directly and deeply involved in developing and writing the standard itself and its associated software and conformance testing suite or are well-known pioneering authorities on HEVC hardware implementation architecture. We hope that this material will help the industry and the community at large to learn how to take full advantage of the promise shown by the new design to facilitate its widespread use.

Chapter 2 by Sjöberg and Boyce describes the high-level syntax of HEVC, which provides a robust, flexible and extensible framework for carrying the coded video and associated information to enable the video content to be used in the most effective possible ways and in many different application environments.

Chapter 3 by Schwarz, Schierl, and Marpe covers the block structures and parallelism features of HEVC, which establish the fundamental structure of its coding design.

Chapter 4 by Lainema and Han describes the intra-picture prediction design in HEVC, which has made it a substantial advance over prior technologies even for still-picture coding.

Chapter 5 by Bross et al. describes inter-picture prediction, which is the heart of what distinguishes a video coding design from other compression applications. Efficient inter-picture prediction is crucial to what makes HEVC powerful and flexible.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00021

Chapter 6 by Budagavi, Fuldseth, and Bjøntegaard describes the transform and quantization related aspects of HEVC. Ultimately, no matter how effective a prediction scheme is applied, there is generally a remaining unpredictable signal that needs to be represented, and HEVC has greater flexibility and adaptivity in its transform and quantization design than ever before, and it also includes some additional coding modes in which the transform stage and sometimes also the quantization stage are skipped altogether.

Chapter 7 by Norkin et al. discusses the in-loop filtering in HEVC, which includes processing elements not found in older video coding designs. As with its AVC predecessor, HEVC contains an in-loop deblocking filter—which has been simplified and made more parallel-friendly for HEVC. Moreover, HEVC introduces a new filtering stage called the sample-adaptive offset (SAO) filter, which can provide both an objective and subjective improvement in video quality.

Chapter 8 by Sze and Marpe covers the entropy coding design in HEVC, through which all of the decisions are communicated as efficiently as possible. HEVC builds on the prior concepts of context-based arithmetic coding (CABAC) for this purpose—pushing ever closer to the inherent entropy limit of efficiency while minimizing the necessary processing requirements, enabling the use of parallel processing, and limiting worst-case behavior.

Chapter 9 by Suzuki et al. covers the compression performance of the design—investigating this crucial capability in multiple ways for various example applications—and including both objective and subjective performance testing. It shows the major advance of HEVC relative to its predecessors. It also shows that the compression improvement cuts across a very broad range of applications, rather than having only narrow benefits for particular uses.

Chapter 10 by Tikekar et al. describes hardware architecture design for HEVC decoding. Decoders are likely to vastly outnumber encoders, and minimizing their cost and power consumption is crucial to widespread use.

Chapter 11 by Tsai, Tsai, and Chen describes hardware architecture design for HEVC encoding. While the requirements for making a decoder are relatively clear—i.e., to properly decode the video according to the semantics of the syntax of the standard—encoders present the open-ended challenge of determining how to search the vast range of possible indications that may be carried by the syntax and select the decisions that will enable good compression performance while keeping within the limits of practical implementation.

We are proud to provide the community with this timely and valuable information collected together into one volume, and we hope it will help spread an understanding of the HEVC standard and of video coding design in general. We expect this book to facilitate the development and widespread deployment of HEVC products and of video-enabled devices and services in general.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00022

References

1. Bossen F, Bross B, Sühring K, Flynn D (2012) HEVC complexity and implementation analysis. IEEE Trans Circuits Syst Video Technol 22(12):1685–1696
2. ITU-T Rec. H.264 and ISO/IEC 14496-10 (2003) Advanced video coding (May 2003 and subsequent editions)
3. ITU-T Rec. H.265 and ISO/IEC 23008-2 (2013) High efficiency video coding. Final draft approval Jan. 2013 (formally published by ITU-T in June, 2013, and in ISO/IEC in Nov. 2013)
4. ITU-T SG16 Q6 and ISO/IEC JTC1/SC29/WG11 (2010) Joint call for proposals on video compression technology. ITU-T SG16 Q6 document VCEG-AM91 and ISO/IEC JTC1/SC29/WG11 document N11113, Kyoto, 22 Jan. 2010
5. Ohm J-R, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards - including High Efficiency Video Coding (HEVC). IEEE Trans Circuits Syst Video Technol 22(12):1669–1684
6. Sullivan GJ, Ohm J-R (2010) Recent developments in standardization of High Efficiency Video Coding (HEVC). In: *Proc. SPIE.* 7798, Applications of Digital Image Processing XXXIII, no. 77980V, Aug. 2010
7. Sullivan GJ, Wiegand T (2005) Video compression - from concepts to the H.264/AVC standard. Proc IEEE 93(1):18–31
8. Sullivan GJ, Ohm J-R, Han W-J, Wiegand T (2012) Overview of the High Efficiency Video Coding (HEVC) standard. IEEE Trans Circuits Syst Video Technol 22(12):1649–1668
9. Wiegand T, Sullivan GJ, Bjøntegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576
10. Wiegand T, Ohm J-R, Sullivan GJ, Han W-J, Joshi R, Tan TK, Ugur K (2010) Special section on the joint call for proposals on High Efficiency Video Coding (HEVC) standardization. IEEE Trans Circuits Syst Video Technol 20(12):1661–1666

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00023

Chapter 2
HEVC High-Level Syntax

Rickard Sjöberg and Jill Boyce

Abstract An HEVC bitstream consists of a sequence of data units called network abstraction layer (NAL) units. Some NAL units contain parameter sets that carry high-level information regarding the entire coded video sequence or a subset of the pictures within it. Other NAL units carry coded samples in the form of slices that belong to one of the various picture types that are defined in HEVC. Some picture types indicate that the picture can be discarded without affecting the decodability of other pictures, and other picture types indicate positions in the bitstream where random access is possible. The slices contain information on how decoded pictures are managed, both what previous pictures to keep and in which order they are to be output. Some NAL units contain optional supplementary enhancement information (SEI) that aids the decoding process or may assist in other ways, such as providing hints about how best to display the video. The syntax elements that describe the structure of the bitstream or provide information that applies to multiple pictures or to multiple coded block regions within a picture, such as the parameter sets, reference picture management syntax, and SEI messages, are known as the "high-level syntax" part of HEVC. A considerable amount of attention has been devoted to the design of the high-level syntax in HEVC, in order to make it broadly applicable, flexible, robust to data losses, and generally highly capable of providing useful information to decoders and receiving systems.

R. Sjöberg (✉)
Ericsson Research, Ericsson, Stockholm, Sweden
e-mail: rickard.sjoberg@ericsson.com

J. Boyce
Vidyo, Inc., Hackensack, NJ, USA
e-mail: jill@vidyo.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_2
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00024

2.1 Introduction

The "high-level syntax" part of HEVC [7, 9] includes the structure of the bitstream as well as signaling of high-level information that applies to one or more entire slices or pictures of a bitstream. For example, the high-level syntax indicates the spatial resolution of the video, which coding tools are used, and describes random access functionalities of the bitstream. In addition to the signaling of syntax elements, the high-level tool decoding processes associated with the syntax elements are also considered to be included in the high level syntax part of the standard. Example high-level syntax decoding processes include reference picture management and the output of decoded pictures.

Figure 2.1 shows an HEVC encoder and decoder. Input pictures are fed to an encoder that encodes the pictures into a bitstream. An HEVC bitstream consists of a sequence of data units called network abstraction layer (NAL) units, each of which contains an integer number of bytes. The first two bytes of a NAL unit constitutes the NAL unit header, while the rest of the NAL unit contains the payload data. Some NAL units carry parameter sets containing control information that apply to one or more entire pictures, while other NAL units carry coded samples within an individual picture.

The NAL units are decoded by the decoder to produce the decoded pictures that are output from the decoder. Both the encoder and decoder store pictures in a decoded picture buffer (DPB). This buffer is mainly used for storing pictures so that previously coded pictures can be used to generate prediction signals to use when coding other pictures. These stored pictures are called reference pictures.

Each picture in HEVC is partitioned into one or multiple slices. Each slice is independent of other slices in the sense that the information carried in the slice is coded without any dependency on data from other slices within the same picture. A slice consists of one or multiple slice segments, where the first slice segment of a slice is called independent slice segment and is independent of other slice segments. The subsequent slice segments, if any, are called dependent slice segments since they depend on previous slice segments.

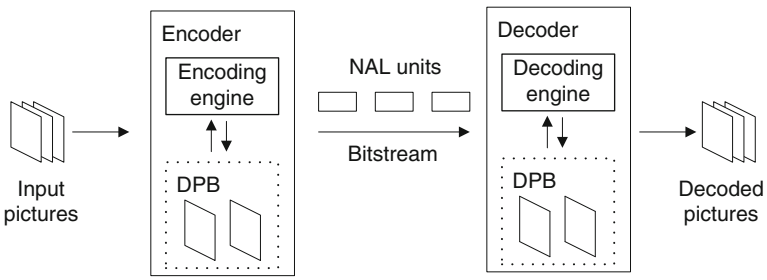


Fig. 2.1 Overview of HEVC encoding and decoding

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00025

Each coded slice segment consists of a slice segment header followed by slice segment data. The slice segment header carries control information for the slice segment, and the slice segment data carries the coded samples. The independent slice header is referred to as the slice header, since the information in this header pertains to all slice segments of the slice.

2.2 The NAL Unit Header and the HEVC Bitstream

There are two classes of NAL units in HEVC—video coding layer (VCL) NAL units and non-VCL NAL units. Each VCL NAL unit carries one slice segment of coded picture data while the non-VCL NAL units contain control information that typically relates to multiple coded pictures. One coded picture, together with the non-VCL NAL units that are associated with the coded picture, is called an HEVC access unit. There is no requirement that an access unit must contain any non-VCL NAL units, and in some applications such as video conferencing, most access units do not contain non-VCL NAL units. However, since each access unit contains a coded picture, it must consist of one or more VCL NAL units—one for each slice (or slice segment) that the coded picture is partitioned into.

2.2.1 The NAL Unit Header

Figure 2.2 shows the structure of the NAL unit header, which is two bytes long. All HEVC NAL unit headers, for both VCL and non-VCL NAL units, start with this two-byte NAL unit header that is designed to make it easy to parse the main properties of a NAL unit; what type it is, and what layer and temporal sub-layer it belongs to.

The first bit of the NAL unit header is always set to '0' in order to prevent generating bit patterns that could be interpreted as MPEG-2 start codes in legacy MPEG-2 systems environments. The next six bits contains the type of the NAL unit, identifying the type of data that is carried in the NAL unit. Six bits means that there are 64 possible NAL unit type values. The values are allocated equally between VCL and non-VCL NAL units so they have 32 types each. NAL unit types will be explained in more detail in Sect. 2.2.2 and 2.2.4.

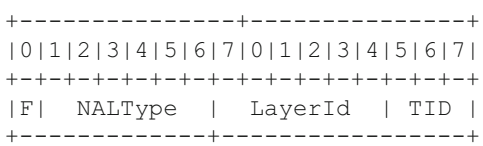


Fig. 2.2 The two-byte NAL unit header

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00026

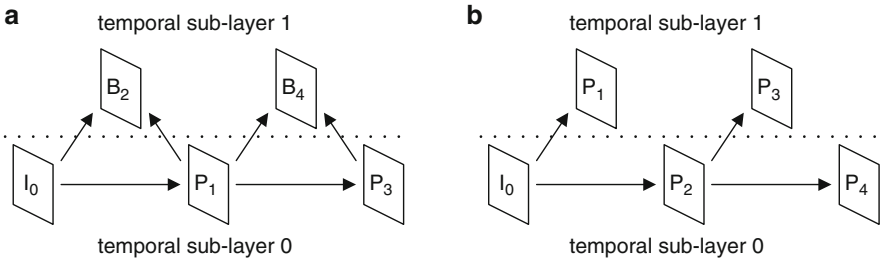


Fig. 2.3 Temporal sub-layer examples

The following six bits contains a layer identifier that indicates what layer the NAL unit belongs to, intended for use in future scalable and layered extensions. Although the first version of HEVC, which was published in June 2013, supports temporal scalability, it does not include any other scalable or layered coding so the layer identifier (layer ID) is always set to '000000' for all NAL units in the first version. In later versions of HEVC, the layer ID is expected to be used to identify what spatial scalable layer, quality scalable layer, or scalable multiview layer the NAL belongs to. These later versions are "layered extensions" to the first version of HEVC and designed to be backwards compatible to the first version [10]. This is achieved by enforcing that all NAL units of the lowest layer (also known as the base layer) in any extension bitstream must have the layer ID set to '000000', and that this lowest layer must be decodable by legacy HEVC decoders that only support the first version of HEVC. For this reason, version one decoders discard all NAL units for which the layer ID is not equal to '000000'. This will filter out all layers except the base layer which can then be correctly decoded.

The last three bits of the NAL unit header contains the temporal identifier of the NAL unit, to represent seven possible values, with one value forbidden. Each access unit in HEVC belongs to one temporal sub-layer, as indicated by the temporal ID. Since each access unit belongs to one temporal sub-layer, all VCL NAL units belonging to the same access unit must have the same temporal ID signaled in their NAL unit headers.

Figure 2.3 shows two different example referencing structures for pictures in a coded video sequence, both with two temporal sub-layers corresponding to temporal ID values of 0 and 1. The slice type is indicated in the figure using I, P, and B, and the arrows show how the pictures reference other pictures. For example, picture B_2 in Fig. 2.3a is a picture using bi-prediction that references pictures I_0 and P_1, see Sect. 2.4.4 for more details on prediction types.

Two very important concepts in order to understand HEVC referencing structures are the concepts of decoding order and output order. Decoding order is the order in which the pictures are decoded. This is the same order as pictures are included in the bitstream and is typically the same order as the pictures are encoded, and is thus also sometimes referred to as bitstream order. There are media transport protocols that allow reordering of coded pictures in transmission, but then the coded pictures

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00027

are reordered to be in decoding order before decoding. The decoding order for the pictures in Fig. 2.3, and other figures in this chapter, is indicated by the subscript numbers inside the pictures.

Output order is the order in which pictures are output from the DPB, which is the order in which the pictures are generally intended to be displayed. Typically, all pictures are output, but there is an optional picture output flag in the slice header that, when set equal to 0, will suppress the output of a particular picture. The HEVC standard uses the term "output" rather than "display" as a way to establish a well-defined boundary for the scope of the standard—what happens after the point of "output" specified in the standard, such as exactly how (and whether) the pictures are displayed and whether any post-processing steps are applied before the display, is considered to be outside the scope of the standard.

Note that pictures that are output are not necessarily displayed. For example, during transcoding the output pictures may be re-encoded without being displayed. The output order of each picture is explicitly signaled in the bitstream, using an integer picture order count (POC) value. The output order of the pictures in each coded video sequence (CVS, see Sect. 2.2.3) is determined separately, such that all output pictures for a particular CVS are output before any pictures of the next CVS that appears in the bitstream in decoding order. Output pictures within each CVS are always output in increasing POC value order. The output order in Fig. 2.3, and other figures in this chapter, is shown by the order of the pictures themselves where pictures are output from left to right. Note that the decoding order and the output order is the same in Fig. 2.3b while this is not the case in Fig. 2.3a. The POC values for pictures in different CVSs are not relevant to each other—only the relative POC relationships within each CVS matter.

HEVC prohibits the decoding process of pictures of any lower temporal sub-layer from having any dependencies on data sent for a higher temporal sub-layer. As shown in Fig. 2.3, no pictures in the lower sub-layer reference any pictures in the higher sub-layer. Since there are no dependencies from higher sub-layers to lower sub-layers, it is possible to remove higher sub-layers from a bitstream to create a new bitstream with fewer pictures in it. The process of removing sub-layers is called sub-bitstream extraction, and the resulting bitstream is called a sub-bitstream of the original bitstream. Sub-bitstream extraction is done by discarding all NAL units which have a temporal ID higher than a target temporal ID value called HighestTid. HEVC encoders are required to ensure that each possible such sub-bitstream is itself a valid HEVC bitstream.

Discarding higher sub-layer pictures can either be done in the path between the encoder and decoder, or the decoder itself may choose to discard higher sub-layers before decoding. One use-case of discarding sub-layers is rate adaptation, where a node in a network between the encoder and decoder removes higher sub-layers when network congestion between itself and the decoder is detected.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00028

Table 2.1 The 32 HEVC VCL NAL unit types

Trailing non-IRAP pictures			
Non-TSA, non-STSA trailing	0	TRAIL_N	Sub-layer non-reference
	1	TRAIL_R	Sub-layer reference
Temporal sub-layer access	2	TSA_N	Sub-layer non-reference
	3	TSA_R	Sub-layer reference
Step-wise temporal sub-layer	4	STSA_N	Sub-layer non-reference
	5	STSA_R	Sub-layer reference
Leading pictures			
Random access decodable	6	RADL_N	Sub-layer non-reference
	7	RADL_R	Sub-layer reference
Random access skipped leading	8	RASL_N	Sub-layer non-reference
	9	RASL_R	Sub-layer reference
Intra random access point (IRAP) pictures			
Broken link access	16	BLA_W_LP	May have leading pictures
	17	BLA_W_RADL	May have RADL leading
	18	BLA_N_LP	Without leading pictures
Instantaneous decoding refresh	19	IDR_W_RADL	May have leading pictures
	20	IDR_N_LP	Without leading pictures
Clean random access	21	CRA	May have leading pictures
Reserved			
Reserved non-IRAP	10–15	RSV	
Reserved IRAP	22–23	RSV	
Reserved non-IRAP	24–31	RSV	

2.2.2 VCL NAL Unit Types

Table 2.1 shows all 32 VCL NAL unit types and their NAL unit type (NALType in Fig. 2.2) values in the NAL unit header. All VCL NAL units of the same access unit must have the same value of NAL unit type and that value defines the type of the access unit and its coded picture. For example, when all VCL NAL units of an access unit have NAL unit type equal to 21, the access unit is called a CRA access unit and the coded picture is called a CRA picture. There are three basic classes of pictures in HEVC: intra random access point (IRAP) pictures, leading pictures, and trailing pictures.

2.2.2.1 IRAP Pictures

The IRAP picture types consist of NAL unit types 16–23. This includes IDR, CRA, and BLA picture types as well as types 22 and 23, which currently are reserved for future use. All IRAP pictures must belong to temporal sub-layer 0 and be coded without using the content of any other pictures as reference data (i.e., using only "intra-picture" or "intra" coding techniques). Note that pictures that are intra coded

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00029

but not marked as IRAP pictures are allowed in a bitstream. The IRAP picture types are used to provide points in the bitstream where it is possible to start decoding. The IRAP pictures themselves are therefore not allowed to be dependent on any other picture in the bitstream.

The first picture of a bitstream must be an IRAP picture, but there may be many other IRAP pictures throughout the bitstream. IRAP pictures also provide the possibility to tune in to a bitstream, for example when starting to watch TV or switching from one TV channel to another. IRAP pictures can also be used to enable temporal position seeking in video content—for example to move the current play position in a video program by using the control bar of a video player. Finally, IRAP pictures can also be used to seamlessly switch from one video stream to another in the compressed domain. This is called bitstream switching or splicing, and it can occur between two live video streams, between a live stream and a stored video file, or between two stored video files. It is always possible to decode from the IRAP picture and onwards to output any subsequent pictures in output order even if all pictures that precede the IRAP picture in decoding order are discarded from the bitstream.

When coding content for storage and later playback or for broadcast applications, IRAP pictures are typically evenly distributed to provide a similar frequency of random access points throughout a bitstream. In real-time communication applications in which random access functionality is not so important or the relatively large number of bits needed to send an IRAP picture is a significant burden that would increase communication delay, IRAP pictures may be very infrequently sent, or may only be sent when some feedback signal indicates that the video data has become corrupted and the scene needs to be refreshed.

2.2.2.2 Leading and Trailing Pictures

A leading picture is a picture that follows a particular IRAP picture in decoding order and precedes it in output order. A trailing picture is a picture that follows a particular IRAP picture in both decoding order and output order. Figure 2.4 shows examples of leading and trailing pictures. Leading and trailing pictures are considered to be associated with the closest previous IRAP picture in decoding order, such as picture I_1 in Fig. 2.4. Trailing pictures must use one of the trailing picture NAL unit types 0–5. Trailing pictures of a particular IRAP picture are not allowed to depend on any leading pictures nor on any trailing pictures of previous IRAP pictures; instead they can only depend on the associated IRAP picture and other trailing pictures of the same IRAP picture. Also, all leading pictures of an IRAP picture must precede, in decoding order, all trailing pictures that are associated with the same IRAP picture. This means that the decoding order of associated pictures is always: (1) The IRAP picture, (2) the associated leading pictures, if any, and then (3) the associated trailing pictures, if any.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00030

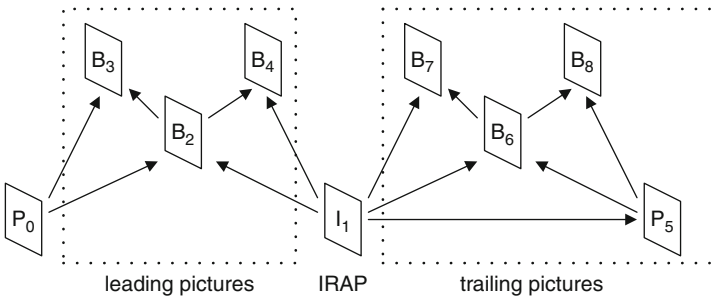


Fig. 2.4 Leading pictures and trailing pictures

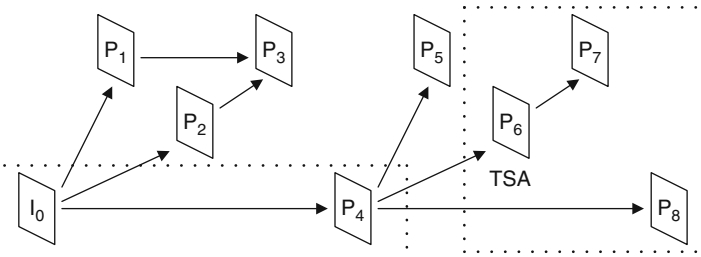


Fig. 2.5 TSA example

There are three types of trailing pictures in HEVC: temporal sub-layer access (TSA) pictures, step-wise temporal sub-layer access (STSA) pictures, and ordinary trailing pictures (TRAIL).

2.2.2.3 Temporal Sub-layer Access (TSA) Pictures

A TSA picture is a trailing picture that indicates a temporal sub-layer switching point. The TSA picture type can only be used for a picture if it is guaranteed that no picture that precedes the TSA picture in decoding order with a temporal ID that is greater than or equal to the temporal ID of the TSA picture itself is used for prediction of the TSA picture or any subsequent (in decoding order) pictures in the same or higher temporal sub-layer as the TSA picture. For example, picture P_6 in Fig. 2.5 can use the TSA picture type since only previous pictures in temporal sub-layer 0 are used for prediction of the TSA picture itself and subsequent pictures in decoding order.

When a decoder is decoding a subset of the temporal sub-layers in the bitstream and encounters a TSA picture type of the temporal sub-layer just above the maximum temporal sub-layer it is decoding, it is possible for the decoder to switch up to and decode any number of additional temporal sub-layers. For the example in Fig. 2.5, a decoder that decodes only temporal sub-layer 0 can from the TSA picture

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00031

either (1) keep decoding temporal sub-layer 0 only, (2) decide to start decoding temporal sub-layer 1 as well as sub-layer 0, or (3) start to decode all three sub-layers. A similar action is possible for a network node that is forwarding only the lowest temporal sub-layer, for example due to a previous network congestion situation. The network node can inspect the NAL unit type of incoming pictures that have a temporal ID equal to 1. This does not require a lot of computational resources since the NAL unit type and the temporal ID are found in the NAL unit header and are easy to parse. When a TSA picture of temporal sub-layer 1 is encountered, the network node can switch to forward any temporal sub-layer pictures succeeding the TSA picture in decoding order without any risk of the decoder not being able to properly decode them as a result of not having all necessary reference pictures that they depend on.

2.2.2.4 Step-wise Temporal Sub-layer Access (STSA) Pictures

The STSA picture type is similar to the TSA picture type, but it only guarantees that the STSA picture itself and pictures of the same temporal ID as the STSA picture that follow it in decoding order do not reference pictures of the same temporal ID that precede the STSA picture in decoding order. The STSA pictures can therefore be used to mark positions in the bitstream where it is possible to switch to the sub-layer with the same temporal ID as the STSA picture, while the TSA pictures can mark positions in the bitstream where it is possible to switch up to any higher sub-layer. One example of an STSA picture in Fig. 2.5 is picture P_2. This picture cannot be a TSA picture since P_3 references P_1. However, picture P_2 can be an STSA picture because P_2 does not reference any picture of sub-layer 1, nor does any sub-layer 1 picture that follows P_2 in decoding order reference any sub-layer 1 picture that precedes P_2 in decoding order. Both TSA and STSA pictures must have a temporal ID higher than 0.

Note also that since prediction from a higher to a lower temporal sub-layer is forbidden in HEVC, it is always possible at any picture to down-switch to a lower temporal sub-layer, regardless of the picture type or temporal sub-layer.

2.2.2.5 Ordinary Trailing (TRAIL) Pictures

Ordinary trailing pictures are denoted with the enumeration type TRAIL. Trailing pictures may belong to any temporal sub-layer. They may reference the associated IRAP picture and other trailing pictures associated with the same IRAP picture, but they cannot reference leading pictures (or any other pictures that are not trailing pictures associated with the same IRAP picture). They also cannot be output after the next IRAP picture in decoding order is output. Note that all TSA and STSA pictures could instead be marked as TRAIL pictures, and that all TSA pictures could be marked as STSA pictures. It is, however, recommended that trailing pictures should use the most restrictive type in order to indicate all possible temporal sub-layer switching points that exist in the bitstream.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00032

2.2.2.6 Instantaneous Decoding Refresh (IDR) Pictures

The IDR picture is an intra picture that completely refreshes the decoding process and starts a new CVS (see Sect. 2.2.3). This means that neither the IDR picture nor any picture that follows the IDR picture in decoding order can have any dependency on any picture that precedes the IDR picture in decoding order. There are two subtypes of IDR pictures, type IDR_W_RADL that may have associated random access decodable leading (RADL) pictures and type IDR_N_LP that does not have any leading pictures. Note that it is allowed, but not recommended, for an encoder to use type IDR_W_RADL even though the IDR picture does not have any leading pictures. It is however forbidden to use type IDR_N_LP for an IDR that has leading pictures. The reason for having two different IDR picture types is to enable system layers to know at random access whether the IDR picture is the first picture to output or not. The POC value of an IDR picture is always equal to zero. Thus, the leading pictures associated with an IDR picture, if any, all have negative POC values.

2.2.2.7 Clean Random Access (CRA) Pictures

A CRA picture is an intra picture that, in contrast to an IDR picture, does not refresh the decoder and does not begin a new CVS. This enables leading pictures of the CRA picture to depend upon pictures that precede the CRA picture in decoding order. Allowing such leading pictures typically makes sequences containing CRA pictures more compression efficient than sequences containing IDR pictures (e.g., about 6 %, as reported in [2]).

Random access at a CRA picture is done by decoding the CRA picture, its leading pictures that are not dependent on any picture preceding the CRA picture in decoding order (see Sect. 2.2.2.8 below), and all pictures that follow the CRA in both decoding and output order. Note that a CRA picture does not necessarily have associated leading pictures.

2.2.2.8 Random Access Decodable Leading (RADL) and Random Access Skipped Leading (RASL) Pictures

The leading pictures must be signaled using either the RADL or RASL NAL unit type. RADL and RASL pictures can belong to any temporal sub-layer, but they are not allowed to be referenced by any trailing picture. A RADL picture is a leading picture that is guaranteed to be decodable when random access is performed at the associated IRAP picture. Therefore, RADL pictures are only allowed to reference the associated IRAP picture and other RADL pictures of the same IRAP picture.

A RASL picture is a leading picture that may not be decodable when random access is performed from the associated IRAP picture. Figure 2.6 shows two RASL pictures which are both non-decodable since picture P_2 precedes the CRA picture in decoding order. Because of its position in decoding order, a decoder that performs

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00033

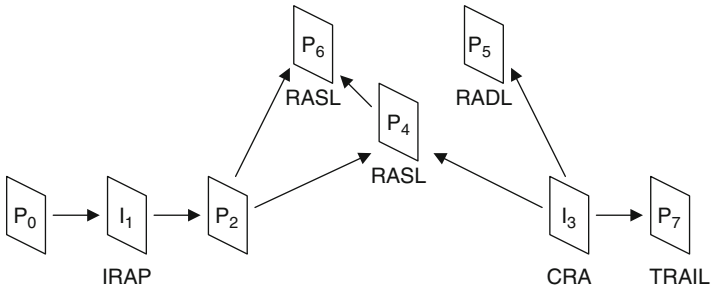


Fig. 2.6 RADL and RASL pictures

random access at the position of the CRA picture will not decode the P_2 picture, and therefore cannot decode these RASL pictures and will discard them. Even though it is not forbidden to use the RASL type for decodable leading pictures, such as the RADL picture in Fig. 2.6, it is recommended to use the RADL type when possible to be more network friendly. Only other RASL pictures are allowed to be dependent on a RASL picture. This means that every picture that depends on a RASL picture must also be a RASL picture. RADL and RASL pictures may be mixed in decoding order, but not in output order. RASL pictures must precede RADL pictures in output order.

All leading pictures of an IDR_W_RADL picture must be decodable and use the RADL type. RASL pictures are not allowed to be associated with any IDR picture. A CRA picture may have both associated RADL and RASL pictures, as shown in Fig. 2.6. RASL pictures are allowed to reference the IRAP picture preceding the associated IRAP picture and may also reference other pictures that follow that IRAP picture in decoding order, but cannot reference earlier pictures in decoding order—e.g., the RASL pictures in Fig. 2.6 cannot reference the picture P_0.

There are three constraints in HEVC that aim to eliminate uneven output of pictures when performing random access. Two of the constraints depend on the variable PicOutputFlag which is set for each picture and indicates whether the picture is to be output or not. This variable is set to 0 when a flag called pic_output_flag is present in the slice header and is equal to 0, or when the current picture is a RASL picture and the associated IRAP picture is the first picture in the CVS (see Sect. 2.2.3). Otherwise PicOutputFlag is set equal to 1.

The first constraint is that any picture that has PicOutputFlag equal to 1 that precedes an IRAP picture in decoding order must precede the IRAP picture in output order. The structure in Fig. 2.7a is forbidden by this constraint, since picture P_1 precedes the CRA in decoding order but follows it in output order. If this was allowed and random access was made at the CRA picture, picture P_1 would be missing, resulting in uneven output.

The second constraint is that any picture that has PicOutputFlag equal to 1 that precedes an IRAP picture in decoding order must precede any RADL picture associated with the IRAP picture in output order. A referencing structure that is

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00034

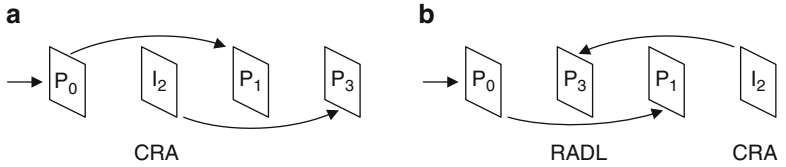


Fig. 2.7 Examples of disallowed referencing structures

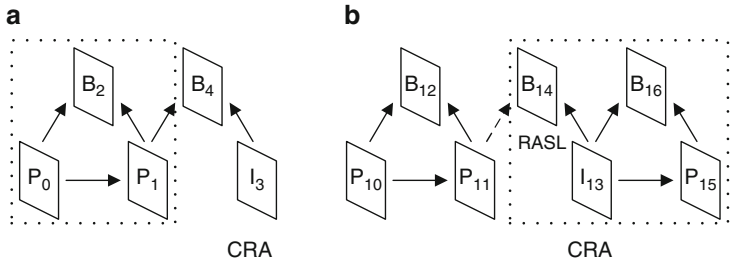


Fig. 2.8 Original (**a**) and new (**b**) referencing structures before splicing has occurred

disallowed by this constraint is shown in Fig. 2.7b since P_1 precedes I_2 but follows P_3 in output order. If this referencing structure was allowed and random access was made at the CRA picture, the missing P_1 picture would cause uneven output.

The third constraint is that all RASL pictures must precede any RADL picture in output order. Since RASL pictures are discarded at random access but RADL are not, any RASL picture that would be displayed after a RADL picture could otherwise potentially cause uneven output upon random access.

2.2.2.9 Splicing and Broken Link Access (BLA) Pictures

Besides using a CRA picture for random access, it is also possible to use a CRA picture for splicing video streams—where a particular IRAP access unit and all subsequent access units of the original bitstream are replaced by an IRAP access unit and the subsequent access units from a new bitstream. The CRA picture is the most compression efficient IRAP picture type so splicing at CRA picture positions may be the most common splicing case.

Figure 2.8a shows an example original bitstream before splicing where the pictures preceding the CRA picture in the bitstream have been highlighted by a dotted box. Figure 2.8b shows an example new bitstream where the IRAP picture and the pictures that follow it in the bitstream are highlighted.

If the CRA picture is followed by RASL pictures, the RASL pictures may not be decodable after splicing since they may reference one or more pictures that are not in the resulting bitstream, e.g. the picture P_{11} in Fig. 2.8b. The decoder should therefore not try to decode those RASL pictures. One way to prevent the decoder from trying

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00035

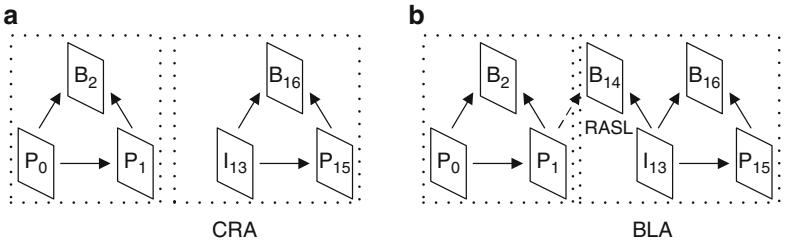


Fig. 2.9 Bitstream after splicing when discarding RASL pictures (**a**) and keeping RASL pictures and converting the CRA picture to BLA (**b**)

to decode these RASL pictures would be to discard them during splicing. The result for splicing the two streams in Fig. 2.8 by discarding RASL pictures is shown in Fig. 2.9a. Note that RADL pictures, if present, could either be kept or discarded.

A disadvantage with this method of discarding RASL pictures is that discarding data in a stream may impact system layer buffers. The splicer may therefore need to be capable of modifying low-level system parameters. If the RASL pictures are forwarded, the system layer buffers are not affected.

Another problem is that the POC values that follow the splicing point would need to indicate the proper output order relationship relative to the pictures that precede the splicing point, since a CRA picture does not begin a new CVS. This could require modification of all POC values that follow the splicing point in the resulting CVS.

An alternative splicing option that is available in HEVC is a "broken link" which indicates that the POC timeline, and the prediction from preceding pictures that RASL pictures may depend on, are broken when splicing is done. Unless the decoder is informed of the broken link, there could be serious visual artifacts if the decoder tries to decode the RASL pictures or if the POC values after the splice point are not appropriately aligned. To avoid visual artifacts, a decoder must be informed when a splicing operation has occurred in order to know whether the associated RASL pictures (if present) should be decoded or not. In HEVC, a broken link access (BLA) picture NAL unit type can be used for such spliced CRA pictures.

During splicing, the CRA picture should be re-typed as a BLA picture. The result of such an operation for the example in Fig. 2.8 is shown in Fig. 2.9b where the RASL picture is kept and the CRA picture is re-typed as a BLA picture. A decoder that encounters BLA and CRA pictures will discard any RASL pictures associated with BLA pictures but decode the RASL pictures associated with CRA pictures. All RADL pictures are required to be decoded.

Like an IDR picture, a BLA picture starts a new CVS and resets the POC relationship calculation. However, the POC value assigned to a BLA picture is not the value 0—instead, the POC value is set equal to the POC value signaled in the slice header of the BLA picture—which is a necessary adjustment since the POC value for a CRA picture would likely be non-zero before its conversion to a BLA

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00036

picture, and changing its POC value to zero would change its POC relationship with other pictures that follow it in decoding order. Note that the BLA picture type is allowed to be used even though no splicing has occurred.

IDR and BLA picture types may look similar, and converting a CRA picture into an IDR picture may look like a possibility during splicing. This is certainly possible but not easy in practice. Firstly, RASL pictures are not allowed to be associated with IDR pictures, so their presence has to be checked before it can be decided whether the IDR picture type actually can be used. Alternatively they can be removed but then it might be necessary to recalculate the buffer parameters. Secondly, the syntax of an IDR picture slice segment header differs for CRA and BLA pictures. One example is that POC information is signaled for CRA and BLA pictures but not for IDR pictures. Therefore the splicer needs to rewrite the slice segment header of the picture. None of this is needed if BLA is chosen, then changing the NAL unit type in the NAL unit headers is sufficient.

As shown in Table 2.1, there are three BLA NAL unit types in HEVC. BLA_N_LP for which leading pictures are forbidden, BLA_W_RADL for which RASL pictures are forbidden but RADL pictures may be present, and BLA_W_LP for which both RASL and RADL pictures are allowed. Even though it is recommended that the splicer checks the subsequent leading picture types and uses the correct BLA type in the spliced output bitstream, it is allowed for a splicer to always use BLA_W_LP. By this, the splicer does not need to inspect the NAL units that follow to check for leading pictures.

2.2.2.10 Sub-layer Reference and Sub-layer Non-reference Pictures

As can be seen in Table 2.1, each leading picture and trailing picture type has two type values. The even picture type numbers indicate sub-layer non-reference pictures and odd picture type numbers indicate sub-layer reference pictures. An encoder can use the sub-layer non-reference picture types for pictures that are not used for reference for prediction of any picture in the same temporal sub-layer. Note that a sub-layer non-reference picture may still be used as a reference picture for prediction of a picture in a higher temporal sub-layer. A network node can use this information to discard individual sub-layer non-reference pictures of the highest sub-layer that it operates on.

Figure 2.10 shows an example where pictures that may use the sub-layer non-reference picture NAL unit types are indicated by an asterisk (*). These are the pictures that are not used for reference by pictures of the same temporal sub-layer, i.e. they do not have an arrow to a picture of the same layer. If HighestTid is two, pictures B_4, B_7, and B_8 may be individually discarded without affecting the ability to decode other pictures in temporal sub-layers up to that sub-layer, but not other pictures. If HighestTid is one, picture B_6 could be similarly discarded, and if HighestTid is zero, picture P_1 could be similarly discarded.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00037

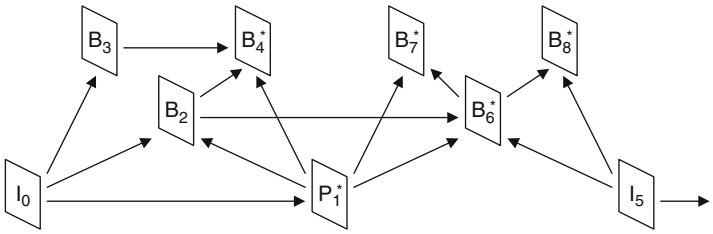


Fig. 2.10 Sub-layer reference and sub-layer non-reference pictures

2.2.2.11 Reserved and Unspecified VCL NAL Unit Types

In addition to the VCL NAL unit types described above, Table 2.1 contains several reserved VCL NAL unit types, which are divided into IRAP and non-IRAP categories. These reserved values are not allowed to be used in bitstreams conforming to the version 1 specification, and are intended for future extensions. Decoders conforming to version 1 of HEVC must discard NAL units with NAL unit types indicating reserved values. Some NAL unit types are also defined as "unspecified", which means they can be used by systems to carry indications or data that do not affect the specified decoding process.

2.2.3 Coded Video Sequences and Bitstream Conformance

A coded video sequence (CVS) in HEVC is a series of access units that starts with an IDR or BLA access unit and includes all access units up to but not including the next IDR or BLA access unit or until the end of the bitstream. A CVS will also start with a CRA access unit if the CRA is the first access unit in the bitstream or if the decoder is set to treat a CRA picture as a BLA picture by external means.

A bitstream is a series of one or more coded video sequences. The bitstream can be in the form of a NAL unit stream, which is a sequence of NAL units in decoding order, or in the form of a byte stream, which is a NAL unit stream with special fixed-value strings called "start codes" inserted in-between the NAL units. The boundaries of the NAL units in a byte stream can be identified by scanning for the start code string values, whereas a NAL unit stream requires some extra framing information to be provided by a system environment in order to identify the location and size of each of the NAL units in the stream.

In order for a bitstream to conform to the HEVC specification, all requirements and restrictions in the HEVC specification must be fulfilled. All syntax restrictions must be met, for example the temporal ID of IRAP NAL units must be equal to 0. Data that does not conform to the HEVC specification can be simply rejected by decoders; the standard does not specify what a decoder should do if such data is encountered. Non-conforming data may be the result of problems in a

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00038

communication system, such as the loss of some of the data packets that contain bitstream data. A decoder may or may not attempt to continue decoding when non-conforming data is encountered. Nevertheless, the output of an HEVC encoder shall always fully conform to the HEVC specification.

There are also syntax element values that are reserved in the specification. These are values that are not in use for a particular version of the HEVC specification, but may be specified and used in future HEVC versions. An encoder is not allowed to use reserved values for a syntax element. If the entire syntax element is reserved, the HEVC specification specifies what value a first version encoder may use. The encoder must obey these rules in order for the output bitstream to be conforming.

A decoder must ignore the reserved syntax element values. If a reserved value is found in the NAL unit header, for instance in the NAL unit type or layer ID syntax elements, the decoder must discard the entire NAL unit. This enables legacy decoders to correctly decode the base layer of any future bitstream that contains additional extension layers that are unknown to the decoders made for earlier versions of the standard.

Some syntax element values are unspecified; those values must be also be ignored by a decoder, as far as their effect on the standard decoding process is concerned. The difference between an unspecified value and a reserved value is that a reserved value may be used in future versions of HEVC while an unspecified value is guaranteed never to be specified in the future and may be used for other purposes that are not defined in the standard. The main purpose of unspecified values is to allow external specifications to make use of them. One example is the unspecified NAL unit type value 48 which is proposed to be used in the HEVC RTP payload specification [11] to signal aggregated packets that contain multiple NAL units. In the proposed RTP payload specification, the value 48 is used as an escape code to indicate that data should not be passed to the HEVC decoder as is, but that additional RTP header data will follow to identify the locations and sizes of the NAL units in the RTP packet. The RTP payload specification is described in more detail in Sect. 2.3.5.

2.2.4 Non-VCL NAL Unit Types

Table 2.2 shows all 32 non-VCL NAL unit types and their NAL unit type values in the NAL unit header.

There are three parameter set types in HEVC, they are explained further in Sect. 2.3.

The access unit delimiter NAL unit may optionally be used to indicate the boundary between access units. If present, the access unit delimiter must signal the same temporal ID as the associated coded picture and be the first NAL unit in the access unit. It has only one codeword in its payload; this codeword indicates what slice types may occur in the access unit.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00039

Table 2.2 The 32 HEVC non-VCL NAL unit types

Non-VCL NAL unit types			
Parameter sets	32	VPS_NUT	Video parameter set
	33	SPS_NUT	Sequence parameter set
	34	PPS_NUT	Picture parameter set
Delimiters	35	AUD_NUT	Access unit delimiter
	36	EOS_NUT	End of sequence
	37	EOB_NUT	End of bitstream
Filler data	38	FD_NUT	Filler data
Supplemental enhancement information (SEI)	39	PREFIX_SEI_NUT	
	40	SUFFIX_SEI_NUT	
Reserved	41–47	RSV	
Unspecified	48–63	UNSPEC	

The end of sequence and end of bitstream types are used to indicate the end of a coded video sequence and the end of a bitstream, respectively. If used, they are placed last in their access units and must indicate temporal layer 0. They have no payload so they both consist of only the two byte NAL unit header.

Filler data NAL units have no impact on the decoding process. The payload is a series of bytes equal to '11111111' followed by a byte equal to '10000000'. It can be used to fill up a data channel to a desired bit rate in the absence of an adequate amount of VCL data. Filler data NAL units must signal the same temporal ID as the coded picture and they are not allowed to precede the first VCL NAL unit in the access unit.

The Supplemental Enhancement Information (SEI) NAL unit type is explained in more detail in Sect. 2.5.

NAL unit types 41–47 are reserved, and types 48–63 are unspecified.

2.3 Parameter Sets

Parameter sets in HEVC are fundamentally similar to the parameter sets in H.264/AVC, and share the same basic design goals—namely bit rate efficiency, error resiliency, and providing systems layer interfaces. There is a hierarchy of parameter sets in HEVC, including the Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) which are similar to their counterparts in AVC. Additionally, HEVC introduces a new type of parameter set called the Video Parameter Set (VPS).

Each slice references a single active PPS, SPS and VPS to access information used for decoding the slice. The PPS contains information which applies to all slices in a picture, and hence all slices in a picture must refer to the same PPS. The slices in different pictures are also allowed to refer to the same PPS. Similarly, the SPS contains information which applies to all pictures in the same coded video sequence. The VPS contains information which applies to all layers within a coded video

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00040

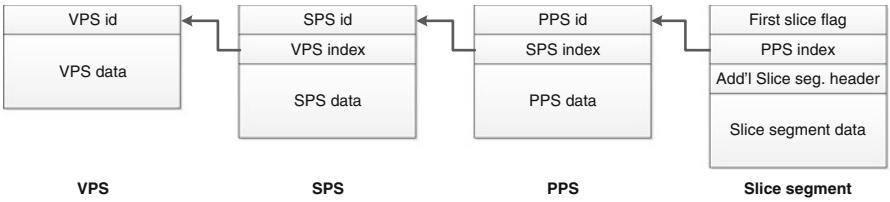


Fig. 2.11 Parameter set referencing hierarchy

sequence, and is intended for use in the upcoming layered extensions of HEVC, which will enable scalable and multiview coding. While the PPS may differ for separate pictures, it is common for many or all pictures in a coded video sequence to refer to the same PPS. Reusing parameter sets is bit rate efficient because it avoids the necessity to send shared information multiple times. It is also loss robust because it allows parameter set content to be carried by some more reliable external communication link or to be repeated frequently within the bitstream to ensure that it will not get lost. This ability to reuse the content of a picture parameter set in different pictures and to reuse the content of SPSs and VPSs in different CVSs is what primarily distinguishes the concept of a "parameter set" from the "picture header" and "sequence header" syntax used in older standards established prior to AVC.

To identify for a given slice the active parameter set at each level of the parameter set type hierarchy, each slice header contains a PPS identifier which references a particular PPS. Within the PPS is an identifier that references a particular SPS. In turn, within the SPS is an identifier that references a particular VPS. Figure 2.11 illustrates this referencing hierarchy.

A parameter set is activated when the current coded slice to be decoded references that parameter set or when an SEI message indicates its activation. All active parameter sets must be available to the decoder when they are first referenced. Parameter sets may be sent in-band or out-of-band, and may be sent repeatedly. Parameter sets may be received in any order. There are no parsing dependencies between parameter sets, e.g. a PPS may be parsed and its parameters stored without the need for referencing the information contained within the SPS to which it refers. These parameter set features provide improved error resiliency, by overcoming some network loss of parameter sets. Additionally, the use of parameter sets allows an individual slice to be decoded even if another slice in the same picture suffered network loss, compared to if a picture header containing the same information was present in a subset of the slices of a picture. The hierarchy of parameter sets can be exploited by systems interfaces, which may benefit from having sequence-level and bitstream-level information available in advance. All parameter sets contain extension flags, to enable backwards-compatible future extensions.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00041

2.3.1 The Video Parameter Set (VPS)

The VPS is a new type of parameter set defined in HEVC, and applies to all of the layers of a bitstream. A layer may contain multiple temporal sub-layers, and all version 1 bitstreams are restricted to a single layer. The future layered extensions under development for scalability and multiview will enable multiple layers, with a backwards compatible version 1 base layer.

AVC did not include a parameter set similar to the VPS, which led to complexity and overhead for AVC's scalable video coding (SVC) and multiview video coding (MVC) extensions. The Scalability Information SEI message in SVC, and the View Scalability Information SEI message in MVC contained some of the same types of information contained within the VPS.

Some of the information contained within the VPS is also duplicated within the SPS. The VPS is required to be included in an HEVC version 1 compliant bitstream or provided through external means, but a version 1 decoder may safely discard NAL units containing a VPS. Duplication of some information in the VPS and SPS requires some (very small degree of) bit rate inefficiency, but enables the VPS to contain information relevant to all layers and their temporal sub-layers, which may be beneficial for systems interfaces, without making it necessary for version 1 decoders to process the VPS to access this information.

Information related to temporal scalability is included in both the VPS and the SPS. The maximum number of temporal sub-layers in the bitstream is indicated. Also a temporal nesting flag which indicates whether temporal sub-layer up-switching can always be performed, e.g. whether all pictures with temporal ID greater than 0 have the TSA picture functionality, as described in Sect. 2.2.2.3. Decoded picture buffer size and picture ordering parameters may be sent for each temporal layer, which indicate restrictions on the size of the decoded picture buffer, and restrictions on the allowable variation of picture decoding and output orders, as described in Sect. 2.4.1. The VPS includes an indication of the maximum number of layers, which for version 1 is restricted to be one. For layered extensions such as scalability and multiview extensions, the relationships between multiple layers are intended to be defined in the VPS extension.

The VPS may also contain layer set and timing information for operation points used for the Hypothetical Reference Decoder, as described in Sect. 2.6.

2.3.2 The Sequence Parameter Set (SPS)

The SPS contains parameters that apply to an entire coded video sequence, and do not change from picture to picture within a coded video sequence. All pictures in the same CVS must use the same SPS. The SPS contains an SPS identifier, as well as an index to identify the associated VPS. The remaining SPS parameters fit into the following categories. Some of the parameters provide key descriptions of the coded

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00042

sequence, which can be useful for systems interfaces. Other parameters describe usage of coding tools, or provide coding tool parameters, which can improve bit rate efficiency. Additionally, the SPS can optionally contain Video Usability Information (VUI) data which provides information that does not directly impact the decoding process, as described in Sect. 2.5.

Key parameters describing the characteristics of the coded sequence are included in the SPS. The profile, tier, and level indications specify conformance points, similar to the profile and level definitions in AVC. A profile defines a set of coding tools, a level imposes capability restrictions on maximum sample rate, picture size, and capabilities of the DPB, etc. HEVC introduces a tier indication, which in combination with level, imposes a maximum bit rate restriction. The coded picture height and width in luma samples are included in the SPS, as well as the conformance window cropping parameters to indicate the output region of the coded picture. Luma and chroma bit depth are also indicated, and their allowable values are constrained by profiles. The SPS also contains some duplicated information from the VPS related to temporal scalability, as described in Sect. 2.3.1.

The SPS also contains parameters to enable or disable coding tools, or to set restrictions on coding tools. In some cases, a coding tool enable flag in the SPS allows a coded slice to avoid containing syntax elements related to the unused coding tool. Examples of tools with enable flags are asymmetric motion partitioning (AMP) as described in Chap. 3, Sample Adaptive Offset (SAO) as described in Chap. 7, and PCM coding as described in Chap. 6. Restrictions on the coding tree block and transform unit sizes are also signaled.

The SPS also can optionally include coding tool parameters, which may also be sent at lower layers if per picture variation is used. These include scaling list data, which provides quantization matrices as described in Chap. 6, and reference picture set (RPS) data as described in Sect. 2.4.2.

2.3.3 The Picture Parameter Set (PPS)

The PPS contains parameters that may change for different pictures within the same coded video sequence. However, multiple pictures may refer to the same PPS, even those with different slice coding types (I, P, and B). Including these parameters within parameter sets rather than within the slice header can improve bit rate efficiency and provide error resiliency when the PPS is transmitted more reliably.

The PPS contains a PPS identifier, as well as an index to a reference SPS. The remaining parameters describe coding tools used in the slices which refer to the PPS. Coding tools are enabled or disabled, including dependent slices, sign data hiding, constrained intra prediction, weighted prediction, trans/quant bypass, tiles, and reference list modification. Coding tool parameters signaled in the PPS include the number of reference indices, initial quantization parameter (QP), and chroma QP offsets. Coding tool parameters may also be signaled in a PPS, e.g. deblocking filter controls, tile configurations, and scaling list data.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00043

For future extensibility, the PPS contains syntax elements indicating extensions to the slice segment header. A syntax element indicates a number of extra slice header bits, in the range of 0–7, to be included at the beginning of the slice header. In addition, a slice segment header extension flag in the PPS indicates the presence of additional bits at the end of the slice segment header.

2.3.4 The Slice Segment Header

The slice segment header contains an index to a reference PPS. The slice segment header contains data identifying the slice segment, including a first slice segment in picture flag and a slice segment address. When dependent slices are used, a slice may be split into multiple slice segments. Some parameters are included only in the first slice segment of a slice, including the slice type (I, P, or B), picture output flag, and long term and short term RPS info (described in more detail in Sect. 2.4.2).

The presence of some coding tool parameters are present in the slice segment header if the tools were enabled in the SPS or PPS, including enabling SAO separately for luma and chroma, enabling deblocking filter operation across slices, and an initial slice quantization parameter (QP) value. Deblocking filter parameters may either be present in the slice segment header or the PPS. If tiles or wavefronts are used, entry points are provided in the slice segment header.

Optional extra slice segment header bits may be included at the beginning of the slice segment header, when indicated in the PPS. Their use is intended for future extensibility, to allow association of parameters with a backwards compatible base layer in a manner that is easily accessible by systems entities.

2.3.5 System Layer Integration Aspects

Virtually all applications of video codecs involve some type of systems interface, for storage and/or transport, and to provide timing information and alignment of the video with other media types, such as audio. Because the HEVC high level syntax design has much commonality with AVC, the systems and transport standards used for carriage of AVC can be updated to carry HEVC [8]. Both HEVC and AVC were designed with network friendliness being a key design goal. The key elements of the HEVC design relevant to the systems interfaces are the NAL unit design, parameter sets, and SEI messages. The main systems standards of interest are Real-time Transport Protocol (RTP) [11], MPEG-2 systems [3, 4], and ISO Base Media File Format (ISOBMFF) [5]. Many other systems and application specifications build upon these three standards. For example, several file format specifications and Dynamic Adaptive Streaming over HTTP (DASH) [6] are based upon ISOBMFF, most broadcast standards use MPEG-2 systems, and most IP videoconferencing applications use RTP.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00044

MPEG-2 transport streams use fixed-length 188-byte packets, which do not directly align with variable length HEVC NAL units. The start of NAL units in MPEG-2 systems can be identified using unique start codes along with the optional byte stream format. Avoidance of start code emulation within NAL units has been part of the NAL unit designs in both AVC and HEVC.

At the time of this writing, a proposed real-time protocol (RTP) payload format for HEVC [11] is in Internet Draft status in the Internet Engineering Task Force (IETF). RTP packets are of variable size, enabling alignment between variable length NAL units and RTP packets. The aggregation packet allows multiple NAL units to be placed within a single RTP packet. For example, parameter sets and/or SEI messages may be placed within the same packet as coded slice segments. The fragmentation unit packet allows a coded slice segment to be split into two or more RTP packets. However, the use of fragmentation units can negatively impact error resiliency, because loss of one packet of a fragmented slice renders the other packets in the slice undecodable. Slices are designed to be individually decodable, with no parsing dependencies between slices. However, parsing of slices does depend upon parameter sets.

In HEVC, like in AVC, parameter sets may be sent in-band or out-of-band, and may be sent repeatedly for improved error resiliency. Many systems specifications provide special handling of parameter sets, to benefit from advance availability of sequence and bitstream level information. For example, when RTP is used, parameter sets may be carried in the Session Description Protocol (SDP), which is used for describing streaming media parameters for session announcement and capabilities exchange/negotiation. In ISOBMFF, parameter sets may be carried in the sample entry of a track.

A Media Aware Network Element (MANE), located in a network between an encoder and decoder, may perform adaptation upon packets of a video transmission, based upon examination of the NAL unit header. As described in Sect. 2.2, all NAL units contain the NAL unit header followed by the NAL unit payload data. The NAL unit header is of fixed length, and contains a temporal identifier value, and reserved bits for a layer identifier to be used in the extensions to HEVC. A MANE may easily access the temporal ID of each packet, and discard those packets whose temporal ID value exceeds a target value, creating a compliant sub-bitstream at a lower frame rate. For systems like MPEG-2 which do not align NAL units with packets, each temporal sub-layer may be placed in a separate Packetized Elementary Streams (PES) to enable sub-bitstream extraction based upon the PES.

The NAL unit header also contains a NAL unit type, which describes the coded picture type. As described in Sect. 2.2.2.1, certain NAL unit type values correspond to random access points. Identification of random access points is particularly important for ISOBMFF. In ISOBMFF, the location of random access points within a bitstream can be identified in multiple ways, such as by the sync sample table and/or some random access points related sample groups.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00045

2.4 Picture Buffering Management

2.4.1 Picture Order Count and the DPB

Every picture in HEVC has a picture order count (POC) value assigned to it, denoted as PicOrderCntVal. It has three main uses: to uniquely identify pictures, to indicate the output position relative to other pictures in the same CVS, and to perform motion vector scaling within the lower-level VCL decoding process. All pictures in the same CVS must have a unique POC value. Pictures from different CVSs may share the same POC value, but pictures can still be uniquely identified since there are no possibilities to mix pictures from one CVS with any picture of another CVS. Gaps in POC values are allowed in a CVS—i.e., the POC value difference between two pictures that are consecutive in output order can differ by more than one (and in fact the amount by which the POC values for consecutive pictures may differ can vary arbitrarily).

The POC value of a picture is signaled by the slice_pic_order_cnt_lsb codeword in the slice header. The range of allowed POC values is from -2^{31} to $2^{31}-1$, so in order to save bits in the slice header, only the least significant bits of the POC value (POC LSB) is signaled. The number of bits to use for POC LSB can be between 4 and 16, and is signaled in the SPS. Since only the POC LSB is signaled in the slice header, the most significant POC value bits (POC MSB) for the current picture are derived from a previous picture, called prevTid0Pic. In order for POC derivation to work the same way even if pictures are removed, prevTid0Pic is set to the closest previous picture of temporal layer 0 that is not a RASL picture, a RADL picture, or a sub-layer non-reference picture. The decoder derives the POC MSB value by comparing the POC value of the current picture with the POC value of the prevTid0Pic picture.

The decoded picture buffer (DPB) in HEVC is a buffer that contains decoded pictures. Decoded pictures other than the current picture may be stored in the DPB either because they are needed for reference, or because they have not been output yet, something that is necessary to enable out-of-order output. Note that the current decoded picture is also stored in the DPB. Figure 2.12 shows two example referencing structures that both need a DPB size of at least three pictures. Pictures P_1 and P_2 in Fig 2.12a both need to be stored in the DPB when P_3 is being decoded since they are both output after P_3. The DPB therefore needs to be capable to store P_1, P_2, and P_3 simultaneously. In Fig. 2.12b, each picture uses two reference pictures so the DPB needs to be large enough to store three pictures simultaneously here as well. The referencing structure in Fig. 2.12b is an example of a so-called low-delay B structure, in which bi-prediction is extensively used without any out-of-order output.

The minimum DPB size that the decoder needs to allocate for decoding a particular bitstream is signaled by the sps_max_dec_pic_buffering_minus1 codeword, which may be sent for each temporal sub-layer in the sequence parameter set. The maximum possible DPB size that is allowed in the first version of HEVC is 16,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00046

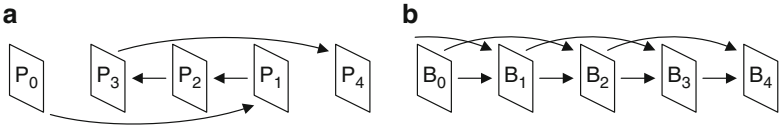


Fig. 2.12 Two referencing structures that need a DPB size of at least three pictures

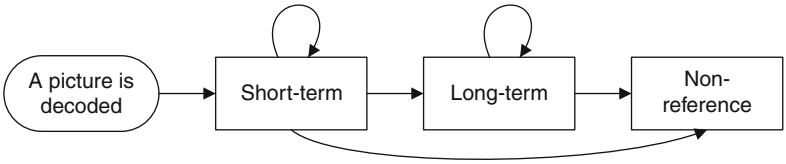


Fig. 2.13 Picture marking

but the maximum size may be further limited depending on the combination of the picture size and the "level" of decoding capability that is used. Note that HEVC specifies the current picture to be included in the DPB, so a DPB size of one would not allow for any reference pictures. If the DPB size is one, all pictures must be intra coded.

The pictures in the DPB are marked to indicate their reference statuses. Each picture in the DPB is marked as "unused for reference", "used for short-term reference", or "used for long-term reference". It is common to refer to these types of pictures as non-reference, short-term, and long-term pictures, respectively.

A reference picture is either a short-term or a long-term picture. The difference is that a long-term picture can be kept in the DPB much longer than a short-term picture. There is a rule determining how long a short-term picture can stay in the DPB. It says that the POC span of the set of pictures consisting of (1) the current picture, (2) prevTid0Pic, (3) the short-term reference pictures in the DPB, and (4) the pictures in the DPB that are waiting for output, must be within half of the POC span covered by POC LSB. This rule guarantees the correctness of POC MSB derivation and improves error robustness by enabling the decoder to identify lost short-term pictures.

A non-reference picture is a picture that is not used for reference but may still be kept in the DPB if it needs to be output later.

Picture markings change for each decoded picture as shown in Fig. 2.13. After a picture has been decoded, it is initially always marked as a short-term picture. A short-term picture may stay as short-term or change to a non-reference or long-term picture. Long-term pictures may stay as long-term or change into non-reference pictures, but they can never be made into short-term pictures again. A non-reference picture can never be made into a reference picture again.

A picture in the DPB may be held for future output regardless of whether it is a reference or non-reference picture. When a picture has been decoded, it is generally waiting for output, unless pic_output_flag in the slice header is equal to 0 or the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00047

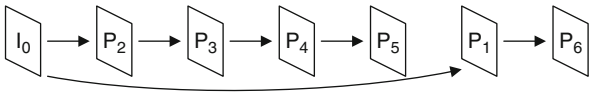


Fig. 2.14 A referencing structure with DPB size equal to 3, NumReorderPics equal to 1, and MaxLatencyPictures equal to 4

picture is a RASL picture that is associated with the first picture in a CVS. If that is the case, the picture will not be output.

The picture marking and picture output are done in separate processes, but when a picture is both a non-reference picture and not waiting for output, the picture storage buffer in the DPB is emptied and can be used to store future decoded pictures. The encoder is responsible to manage picture marking and picture output such that the number of pictures in the DPB does not exceed the DPB size as indicated by sps_max_dec_pic_buffering_minus1.

Two other codewords in the SPS that are related to picture output are sps_max_num_reorder_pics and sps_max_latency_increase_plus1, which both can be sent for each temporal sub-layer. sps_max_num_reorder_pics, here denoted as NumReorderPics, indicates the maximum number of pictures that can precede any picture in decoding order and follow it in output order. The value of NumReorderPics for the referencing structure in Fig. 2.12a is two since picture P_3 has two pictures that precede it in decoding order but follow it in output order. NumReorderPics for Fig. 2.12b is zero since no picture is sent out-of-order.

sps_max_latency_increase_plus1 is used to signal MaxLatencyPictures, which indicates the maximum number of pictures that can precede any picture in output order and follow that picture in decoding order. Figure 2.14 shows the difference between NumReorderPics and MaxLatencyPictures. NumReorderPics is here equal to one since P_1 is the only picture that is sent out-of-order. MaxLatencyPictures is set to four since pictures P_2, P_3, P_4, and P_5 all precedes P_1 in output order. The minimum DPB size for this referencing structure is three.

One can say that NumReorderPics denotes the minimum number of picture stores in the DPB that are necessary for taking care of out-of-order pictures, and that MaxLatencyPictures denotes the minimum amount of coding delay, measured in pictures, that is caused by out-of-order pictures.

For low-delay applications it is recommended to use referencing structures that have no coding delay caused by out-of-order output. This is achieved by ensuring that the decoding order and output order is the same which can be expressed by signaling either NumReorderPics or MaxLatencyPictures, or both, equal to zero.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00048

Table 2.3 The five RPS lists

List name	Long-term or short-term	Availability flag	POC
RefPicSetStCurrBefore	Short-term	Available	Lower
RefPicSetStCurrAfter	Short-term	Available	Higher
RefPicSetStFoll	Short-term	Unavailable	–
RefPicSetLtCurr	Long-term	Available	–
RefPicSetLtFoll	Long-term	Unavailable	–

2.4.2 Reference Picture Sets

The process of marking pictures as "used for short-term reference", "used for long-term reference", or "unused for reference" is done using the reference picture set (RPS) concept. An RPS is a set of picture indicators that is signaled in each slice header and consists of one set of short-term pictures and one set of long-term pictures. After the first slice header of a picture has been decoded, the pictures in the DPB are marked as specified by the RPS.

The pictures in the DPB that are indicated in the short-term picture part of the RPS are kept as short-term pictures. The short-term or long-term pictures in the DPB that are indicated in the long-term picture part in the RPS are converted to or kept as long-term pictures. And finally, pictures in the DPB for which there is no indicator in the RPS are marked as unused for reference. Thus, all pictures that have been decoded that may be used as references for prediction of any subsequent pictures in decoding order must be included in the RPS.

An RPS consists of a set of picture order count (POC) values that are used for identifying the pictures in the DPB. Besides signaling POC information, the RPS also signals one flag for each picture. Each flag indicates whether the corresponding picture is available or unavailable for reference for the current picture. Note that even though a reference picture is signaled as unavailable for the current picture, it is still kept in the DPB and may be made available for reference later on and used for decoding future pictures. From the POC information and the availability flag, five lists of reference pictures as shown in Table 2.3 can be created.

The list RefPicSetStCurrBefore consists of short-term pictures that are available for reference for the current picture and have POC values that are lower than the POC value of the current picture. RefPicSetStCurrAfter consist of available short-term pictures with a POC value that is higher than the POC value of the current picture. RefPicSetStFoll is a list that contains all short-term pictures that are made unavailable for the current picture but may be used as reference pictures for decoding subsequent pictures in decoding order. Finally, the lists RefPicSetLtCurr and RefPicSetLtFoll contain long-term pictures that are available and unavailable for reference for the current picture, respectively.

Figure 2.15 and Table 2.4 show an example referencing structure using three temporal sub-layers and the content of the RPS lists for each picture in decoding order.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00049

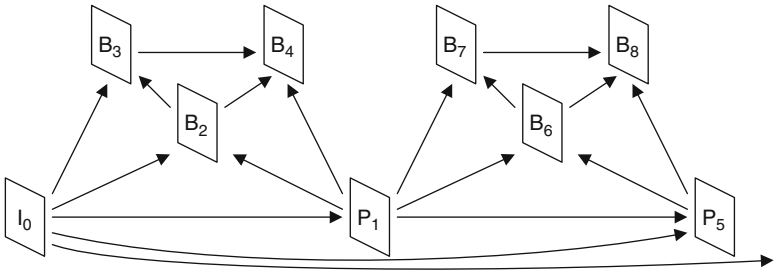


Fig. 2.15 Example referencing structure

Table 2.4 RPS for each picture of Fig. 2.15

Picture	POC	RefPicSetStCurr	RefPicSetStFoll	RefPicSetLtCurr	RefPicSetLtFoll
I_0	0	–	–	–	–
P_1	4	I_0	–	–	–
B_2	2	I_0, P_1	–	–	–
B_3	1	I_0, B_2	P_1	–	–
B_4	3	P_1, B_2, B_3	–	–	I_0
P_5	8	P_1	–	I_0	–
B_6	6	P_1, P_5	–	–	I_0
B_7	5	P_1, B_6	P_5	–	I_0
B_8	7	P_5, B_7	–	–	I_0

An IDR picture resets the codec which includes turning all pictures in the DPB into non-reference pictures. Since the RPSs of IDR pictures are empty, there is no RPS syntax signaled for IDR pictures. All lists in Table 2.4 are therefore empty for the IDR picture I_0. At picture B_3, picture P_1 is put in RefPicSetStFoll since P_1 is not referenced by B_3. P_1 is however kept in the DPB since it is used for future pictures. At picture B_4, I_0 is made into a long-term picture in this example, it is therefore put in RefPicSetLtFoll since it is not referenced by B_4. At picture P_5, the encoder makes pictures B_2 and B_3 non-reference pictures by not including them in the RPS at all. At the same time, picture I_0 is moved to RefPicSetLtCurr since it is referenced by P_5. I_0 is thereafter kept in RefPicSetLtFoll for later use.

The encoder is required to ensure that every picture that is indicated in RefPicSetStCurr and RefPicSetStFoll are present in the DPB. If this is not the case, the decoder should infer that as a bitstream error and take appropriate action. However, if there is no corresponding picture in the DPB for an entry in the RPS that is indicated not to be used for reference for the current picture, the decoder takes no action since this situation may occur due to the removal of individual sub-layer non-reference pictures or entire higher temporal sub-layers.

Although no RPS is sent for IDR pictures, both CRA and BLA pictures may have pictures in their RPS so that the associated RASL pictures can use those

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00050

pictures in the RPS as references for prediction. However, it is required that the RefPicSetStCurr and RefPicSetLtCurr lists are both empty for CRA and BLA pictures.

2.4.3 Reference Picture Set Signaling

Three pieces of information are signaled for each picture in the RPS; the POC information, the availability state, and whether the picture is a short-term or long-term picture.

The POC information for short-term pictures is signaled in two groups. Group S0 is signaled first and it consists of all short-term pictures with lower POC values than the current picture. This group is followed by group S1 which contains all short-term pictures with higher POC values than the current picture. The information for each group is sent in POC distance order relative to the current picture, starting with the POC value that is closest to the POC value of the current picture. For each picture, a POC delta relative to the previous picture is signaled. The current picture acts as the previous picture for the first picture in each group since there is no previous picture for the first pictures.

Coding long-term pictures by POC deltas may result in very long codewords since long-term pictures can stay in the DPB for a very long time. Therefore, long-term pictures are instead signaled by their POC LSB values. The same number of bits that is used for the slice header POC LSB values is used also for long-term pictures. The decoder will match each POC LSB value signaled in the RPS with the POC LSB values of the pictures in the DPB. Since it is possible to have more than one picture with the same POC LSB value in the DPB, there is an optional possibility to also signal POC MSB information for long-term pictures. This MSB information must be sent when there is a risk that a decoder is unable to correctly identify the pictures. One way of avoiding this risk is to always signal POC MSB information for long-term pictures when the corresponding POC LSB value has been used by two or more different long-term pictures.

The availability state for each picture in the RPS is signaled by a one-bit flag where '1' indicates that the picture is available for reference for the current picture and '0' indicates that it is not.

Table 2.5 shows example RPS syntax for picture B_7 in Fig. 2.15. The first column in the table shows whether the syntax is related to signaling short-term pictures or long-term pictures. The second column contains the HEVC specification name of each syntax element. The third column indicates the related pictures in the RPS, and the fourth column shows the value of the syntax element in the example. The fifth column shows the type of the syntax element where 'uvlc' denotes a universal variable-length code, 'flag' is a one-bit binary flag, and 'flc' is a fixed-length code. The last column shows the resulting bits for encoding the value of each syntax element using the type.

The POC value of picture B_7 in Fig. 2.15 is 5 and its RPS contains three short-term pictures; picture P_1 in group S0 and pictures B_6 and P_5 in group S1. The first

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00051

Table 2.5 RPS syntax for picture B_7 in Fig. 2.15

Part	Syntax element	Picture	Value	Type	Codeword
Short-term	num_negative_pics	P_1	1	uvlc	'010'
	num_positive_pics	P_5, B_6	2	uvlc	'011'
	delta_poc_s0_minus1	P_1	0	uvlc	'1'
	used_by_curr_pic_s0_flag		Used	flag	'1'
	delta_poc_s1_minus1	B_6	0	uvlc	'1'
	used_by_curr_pic_s1_flag		Used	flag	'1'
	delta_poc_s1_minus1	P_5	1	uvlc	'010'
	used_by_curr_pic_s1_flag		Unused	flag	'0'
Long-term	num_long_term_pics	I_0	1	uvlc	'010'
	poc_lsb_lt		0	flc	'00000000'
	used_by_curr_pic_lt_flag		Unused	flag	'0'
	delta_poc_msb_present_flag		0	flag	'0'

two syntax elements for the RPS convey the number of pictures in S0 and S1; this is encoded by the uvlc codes '010' and '011'. Then the picture in group S0 is signaled. The POC delta of P_1 is 1 since its POC value is 4 and the current picture has a POC value equal to 5. POC deltas are subtracted by 1 before encoding, so the final signaled value is 0 which results in the uvlc code '1'. Picture P_1 is used for reference for the current picture B_7; this is signaled by the flag '1'.

The next syntax elements cover the pictures in S1. The current picture has a POC value equal to 5 so the codeword '1' is used for B_6 since its POC value is equal to 6. Picture B_6 is also used for reference for the current picture which is indicated by the flag '1'. The second picture in S1 is P_5 which has a POC value of 8. It is coded relative to the previous picture in S1 which is B_6 with POC value equal to 6. The delta is 2 which subtracted by one is equal to 1 and encoded as '010'. Picture P_5 is not used for reference by the current picture B_7; this is signaled by the flag '0'.

The next codeword in the RPS signals the number of long-term pictures in the RPS, which is one and uvlc coded using '010'. Then the POC LSB value of the long-term picture I_0 is signaled. This is by the codeword '00000000' assuming that 8 bits are used for signaling POC LSB values. The long-term picture I_0 is not used for reference by the current picture B_7; this is signaled with the flag '0'. Finally, the POC MSB information is not signaled for this long-term picture since there are no other pictures in the DPB sharing the same POC LSB value; this is signaled with the present flag '0'.

2.4.3.1 RPS Signaling in the Slice Header and SPS

The RPS information is signaled in every slice header for resilience reasons, but repeating the RPS information as is for every slice could cost many bits. Not only can pictures be split into multiple slices for which the RPS must be repeated, the pictures in a bitstream are often coded by repeating the same GOP structure, so the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00052

same RPS information is repeated for the pictures that share the same position in the GOP structure.

To exploit the redundancy and reduce the overall bit cost, some of the RPS syntax can be sent in the SPS and referred to from the slice header. The short-term picture part of an RPS is coded relative to the POC value of the current picture. This makes it possible to store multiple short-term RPS parts in a list in the SPS and only signal a list index in the slice header. The list may contain one RPS for each picture position in the GOP structure. The number of RPSs in the SPS would then be equal to the GOP length, and each slice would only need to send a list index in order to signal its short-term picture part.

Sending the RPS information for each GOP position may require relatively many bits in the SPS, there are examples of several hundred bits for some GOP structures. In order to save bits in the SPS, there is the option in HEVC to use RPS prediction. This requires the RPSs to be sent in GOP decoding order and exploits the fact that each RPS is similar to the previous one. Picture references may be removed from one RPS to the next in decoding order, but no more than one new picture is possible to add relative to the previous RPS. The RPS prediction mechanism in HEVC is utilizing this property, and reductions of up to 50 % of the RPS bit count in the SPS can be achieved relative to explicit signaling.

2.4.4 Reference Picture Lists

There are two types of sample prediction in HEVC, intra prediction and inter prediction. Intra prediction does not include prediction from any reference picture, only sample prediction using reconstructed samples of the same picture is allowed. Inter prediction uses reference pictures where picture identifiers, called reference indices, and motion vectors are used to specify what part of which reference picture to use for prediction.

There are three slice types in HEVC. The intra (I) slice for which only intra prediction is allowed. The predictive (P) slice which in addition to intra prediction also allows inter prediction from one reference picture per block using one motion vector and one reference index. This is called uni-prediction. Finally, there is the bi-predictive (B) slice which in addition to intra and uni-prediction also allows inter prediction using two motion vectors and two reference indices. This results in two prediction blocks that are combined to form a final prediction block. Using bi-prediction is generally more compression efficient than using uni-prediction but the computational complexity is higher.

Note that the use of uni-prediction and bi-prediction is completely decoupled from both the output order as well as from the number of reference pictures used. Different blocks within the same P slice can reference different reference pictures, and a block that uses bi-prediction can have both its motion vectors pointing to the same picture.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00053

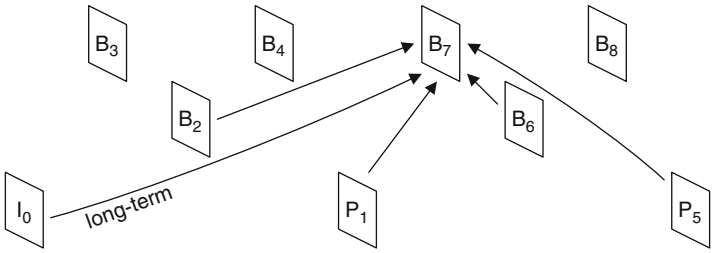


Fig. 2.16 Reference picture list construction example

When a slice header of a P or B slice has been decoded, the decoder sets up reference picture lists for the current slice. There are two reference picture lists, L0 and L1. L0 is used for both P and B slices while L1 is only used for B slices. Figure 2.16 shows a reference picture list example for a picture B_7 that uses five reference pictures for prediction.

First, temporary lists for L0 and L1 are constructed. Temporary list L0 starts with the pictures in RPS list RefPicSetStCurrBefore sorted in descending POC value order. These are all short-term pictures that are available for reference for the current picture and have a POC value that is less than the POC value of the current picture. The temporary list L0 for picture B_7 in Fig. 2.16 therefore starts with P_1 followed by B_2. These are followed by the pictures in the RPS list RefPicSetStCurrAfter sorted in ascending POC value order, which are B_6 and P_5 in Fig. 2.16. Finally, the available long-term pictures are added, so picture I_0 is the last picture to be added. The final temporary list L0 for picture B_7 is then $\{P_1, B_2, B_6, P_5, I_0\}$. The temporary list L1 is set up similar to L0, but the order of RefPicSetStCurrBefore and RefPicSetStCurrAfter is swapped. The temporary list L1 for picture B_7 therefore becomes $\{B_6, P_5, P_1, B_2, I_0\}$.

The temporary lists are used for constructing the final L0 and L1 lists. The length of L0 and L1 are signaled in the PPS, but can be overridden by optional codewords in the slice header. The maximum list length is 15. If the specified length of L0 or L1 is less than the length of the temporary lists L0 or L1, respectively, the final list is constructed by truncating the corresponding temporary list. If the specified length is greater than the length of the respective temporary list, the pictures are repeatedly included from the temporary list. For example, L0 becomes equal to $\{P_1, B_2\}$ if the length is 2 and equal to $\{P_1, B_2, B_6, P_5, I_0, P_1, B_2, B_6, P_5\}$ if the length is 9. The reason for enabling a list that is longer than the temporary list is weighted prediction, to enable different weighting factors to be applied to the same reference picture.

An alternative method to construct L0 and L1 from the temporary lists is through explicit list signaling. In this case, there is a codeword for each entry in the list that specifies which picture from the temporary list to use. So if the length of L0 is specified to be equal to three, there are three codewords for specifying L0. For example, if those three codewords are equal to 1, 1, 0 the list L0 becomes $\{B_2, B_2, P_1\}$ when P_1 is the first picture in temporary list L0 and B_2 is the second.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00054

The final lists L0 and L1 are used for motion compensation. For uni-prediction, one motion vector and one reference picture index is indicated for a block. For instance, the first picture in L0 is the picture to use for motion compensation if the signaled index for a block is equal to 0.

2.5 Video Usability Information (VUI) and Supplemental Enhancement Information (SEI)

HEVC VUI and SEI are similar in behavior to the VUI and SEI of AVC. The video usability information (VUI) is an optional section of syntax within the SPS. VUI data do not directly impact the decoding process, but provides useful information in two main categories. The first category is related to display of the decoded pictures, including information such as aspect ratio, overscan, and color primaries. The category also includes timing information which is used by the hypothetical reference decoder (HRD), as described in Sect. 2.6. The second category of VUI data is for bitstream restrictions, to provide information to the decoder indicating that the encoder did not exercise the full flexibilities of the standard. The restrictions signaled relate to tiles, motion vectors, reference picture lists, and bytes per coded picture.

SEI messages provide metadata and are generally optional. SEI messages are carried within SEI NAL units. In AVC, all SEI messages are considered to be of a prefix type, which means that the SEI message is required to precede all VCL NAL units of an access unit. HEVC introduces the concept of a suffix SEI message, which follows a VCL NAL unit of an access unit. Table 2.6 lists all of the HEVC SEI messages, and indicates whether they are of prefix or suffix type. Some messages may be used in either prefix or suffix mode.

SEI messages have different rules for persistence. Some SEI messages apply only to the current access unit, while others persist until another SEI message cancels or replaces it or until a new CVS begins. The scalable nesting SEI message can be used to indicate if an SEI message applies to particular temporal sub-layers, and in the future layered extensions, to which particular layers.

Many of the SEI messages are very similar to related messages in AVC, although they do not always follow the exact same syntax. Buffering period, picture timing, and decoding unit info messages are used in the HRD operation, as described in Sect. 2.6. The following SEI messages are carried over from AVC: pan-scan rectangle, filler payload user data registered, user data unregistered, recovery point, scene information, progressive refinement segment start, progressive refinement segment end, film grain characteristics, post filter hint, tone mapping information, and frame packing arrangement.

In AVC, the scalable nesting SEI message is part of the SVC extension in Annex G, but in HEVC it is included in version 1, because of version 1 support for temporal scalability. The HEVC temporal sub-layer zero information SEI message is similar

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00055

Table 2.6 HEVC SEI messages

SEI message	Type	Description
Buffering period	Prefix	Provides parameters for HRD initialization
Picture timing	Prefix	Provides HRD parameters and interlaced picture indication
Pan scan rectangle	Prefix	Provides conformance cropping window parameters, to indicate when output pictures are smaller than decoded pictures
Filler payload	Prefix/suffix	Carries unused data, to enable encoder to achieve desired bit rate
User data registered	Prefix/suffix	Carries user-specific data, with type registered through registration authority
User data unregistered	Prefix/suffix	Carries user-specific data, not registered
Recovery point	Prefix	Indicates first picture with acceptable quality after non-IRAP random access
Scene info	Prefix	Description of scene and scene transition
Picture snapshot	Prefix	Indicates picture intended for use as still-image snapshot of the video
Progressive refinement seg. start	Prefix	Indicates start of a sequence of coded pictures to progressively improve quality of a particular picture
Progressive refinement seg. end	Prefix	Indicates end of a sequence of coded pictures to progressively improve quality of a particular picture
Film grain characteristics	Prefix	Describes a parameterized model for film grain synthesis
Post filter hint	Prefix/suffix	Provides coefficients of a post filter
Tone mapping info	Prefix	Provides remapping information of the colour samples of the output pictures for customization to particular display environments
Frame packing arrangement	Prefix	Indicates that the output picture contains multiple distinct spatially packed frames, and the particular arrangement used
Display orientation	Prefix	Indicates to decoder to rotate or flip the output picture prior to display
Structure of pictures info	Prefix	Provides series pattern information of coded picture types
Decoded picture hash	Suffix	Provides a hash for each colour component of the decoded picture, to assist decoder to detect mismatch with encoder
Active parameter sets	Prefix	Indicates the active VPS and SPS

(continued)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00056

Table 2.6 (continued)

SEI message	Type	Description
Decoding unit info	Prefix	Provides HRD parameters for sub-AU decoding units
Temporal sub layer zero index	Prefix	Provides information to assist decoder to detect missing coded pictures
Scalable nesting	Prefix	Associates other SEI messages with bitstream subsets
Region refresh info	Prefix	Indicates if slice segments belong to a refreshed region of the picture
Reserved	Prefix/suffix	For future extensions

to the temporal level zero dependency representation index SEI message in AVC's SVC extension. Some of the newly introduced SEI messages in HEVC are described below.

The decoded picture hash SEI provides a calculated hash for each colour component of the decoded picture. Three different hash calculation methods are supported: MD5, CRC, and checksum. This SEI message is intended for debugging and interoperability testing, and allows a decoder to determine if the decoded picture exactly matches that of the encoder.

The display orientation SEI message informs the decoder of a recommended transformation to be applied to the cropped decoded picture prior to display. The message includes indications to flip the picture horizontally or vertically, and an anticlockwise rotation amount.

The structure of pictures SEI message provides information about patterns of coded pictures within the coded video sequence. Patterns of pictures in terms of their values of NAL unit type, temporal ID values, short-term reference picture set index, and POC delta.

The active parameter sets SEI message provides easily accessible information to a middle box or decoder of the active video parameter set ID value, and optionally sequence parameter set ID values. Without using this SEI message, it is required to parse the slice header to find the PPS ID, access the contents of the picture parameter set to find the SPS ID, and access the contents of the sequence parameter set to determine the VPS ID.

2.6 Hypothetical Reference Decoder (HRD)

The operation of the HRD in HEVC [1] behaves similarly to the HRD of H.264/AVC, but provides additional functionalities. The HRD allows an encoder to specify the constraints of a bitstream to identify the capabilities needed to ensure that the bitstream can be correctly buffered, decoded, and output. Signaling of HRD parameters is optional for an encoder. HRD parameters do not directly

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00057

impact decoding operation, but when used, must conform to profile, tier, and level constraints. The HRD models a coded picture buffer (CPB) using a leaky bucket model with the following parameters: transmission bit rate R, buffer size B, and initial buffer fullness F. In addition to access unit level HRD operation, as provided in H.264/AVC, the HEVC HRD adds support for sub-picture level HRD operation for ultra-low delay applications, by specifying parameters per Decoding Unit (DU).

HEVC version 1 includes temporal scalability capability and the planned extensions will support multiple layers, for spatial or quality scalability and for multi-view, with a backwards-compatible base layer. Even though HEVC version 1 itself does not include multi-layer capability, HRD parameters may be specified for multiple layers. This allows an HEVC version 1 decoder to receive HRD parameters of a multi-layer bitstream, even though the decoder is only capable of decoding the base layer.

To provide multi-layer HRD functionality, one or more sets of HRD parameters may be sent, each corresponding to an operation point. An operation point defines the parameters used for sub-bitstream extraction, which include a list of target layers and a target highest temporal layer. Multiple operation points may be specified for a particular bitstream.

A subset of the HRD parameters is signaled in the SPS or VPS. These include indications if HRD information is included in the bitstream, and if the HRD parameters apply to either a Type I bitstream, which includes only VCL NAL units and filler data NAL units, or to a Type II bitstream, which also includes other types of non-VCL NAL units. Access unit and/or decoding unit granularity is also indicated. Fixed frame rate can also be specified.

The remaining HRD parameters are signaled in the buffering period, picture timing, and decoding unit SEI messages, as follows. The buffering period SEI message includes initial CPB and DPB delay and offset parameters which are used for deriving the nominal CPB and DPB removal time of access units. The buffering period SEI message may also optionally include alternate values for those parameters when leading pictures (as described in Sect. 2.2.2.2) are discarded. When used, buffering period SEI messages are required to be sent for each IRAP access unit, to enable random access.

The picture timing SEI message is typically sent for each coded picture when the HRD is used, and it includes parameters for CPB removal delay and DPB output delay for each picture. When sub-picture operation is used for ultra-low delay operation, CPB removal delay and DPB output delay for each DU may either be sent in the picture timing SEI message for all DUs in the picture, or may be sent in a separate decoding unit info SEI message for each DU.

References

1. Deshpande S, Hannuksela MM, Kazui K, Schierl T (2013) An improved hypothetical reference decoder for HEVC. In Proc. SPIE. 8666, Visual Information Processing and Communication IV, no. 866608, Feb. 2013

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00058

2. Fujibayashi A, Tan TK (2011) Random access support for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D234, Daegu, Jan. 2011
3. Grüneberg K, Schierl T, Narasimhan S (2013) ISO/IEC 13818-1:2013/FDAM 3, Transport of High Efficiency Video Coding (HEVC) video over MPEG-2 systems
4. ISO/IEC 14496-15:2012/DAM 2, Carriage of HEVC
5. ISO/IEC 14496-15:2012, Information technology - coding of audio-visual objects - Part 12: ISO base media file format
6. ISO/IEC 23009-1:2012 Information technology - dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats
7. ITU-T Rec. H.265 and ISO/IEC 23008-2 (2013) High efficiency video coding
8. Schierl T, Hannuksela MM, Wang Y-K, Wenger S (2012) System layer integration of high efficiency video coding. IEEE Trans Circuits Syst Video Technol 22(12):1871–1884
9. Sjöberg R, Chen Y, Fujibayashi A, Hannuksela MM, Samuelsson J, Tan TK, Wang Y-K, Wenger S (2012) Overview of HEVC high-level syntax and reference picture management. IEEE Trans Circuits Syst Video Technol 22(12):1858–1870
10. Sullivan GJ, Boyce JM, Chen Y, Ohm J-R, Segall CA, Vetro A (2013) Standardized extensions of High Efficiency Video Coding (HEVC). IEEE J Sel Top Signal Process 7(6):1001–1016
11. Wang Y-K, Sanchez Y, Schierl T, Wenger S, Hannuksela MM (2013) RTP payload format for high efficiency video coding. http://tools.ietf.org/html/draft-ietf-payload-rtp-h265-03

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00059

Chapter 3
Block Structures and Parallelism Features in HEVC

Heiko Schwarz, Thomas Schierl, and Detlev Marpe

Abstract In block-based hybrid video coding, each picture is partitioned into blocks of samples and multiple blocks within a picture are aggregated to form slices as independently decodable entities. While adhering to this basic principle, the new High Efficiency Video Coding (HEVC) standard provides a number of innovative features both with respect to sample aggregating block partitioning and block aggregating picture partitioning. This chapter first describes the quadtree-based block partitioning concept of HEVC for improved prediction and transform coding, including its integral parts of coding tree blocks (CTBs), coding blocks (CBs), prediction blocks (PBs), and transform blocks (TBs). Additionally, the coding efficiency improvements for different configurations of HEVC with respect to the choice of different tree depths and block sizes for both prediction and transform are evaluated. As one outcome of this experimental evaluation, it was observed that more than half of the average bit-rate savings of HEVC relative to its predecessor H.264 | MPEG-4 AVC can be attributed to its increased flexibility of block partitioning for prediction and transform coding. The second part of this chapter focuses on improved picture partitioning concepts for packetization and parallel processing purposes in HEVC. This includes the discussion of novel tools for supporting high-level parallelism, such as tiles and wavefront parallel processing (WPP). Furthermore, the new concept for fragmenting slices into dependent slice segments for both parallel bitstream access and ultra-low delay processing is presented along with a summarizing discussion of the pros and cons of both WPP and tiles.

H. Schwarz (✉) • T. Schierl • D. Marpe
Fraunhofer HHI, Einsteinufer 37, Berlin, Germany
e-mail: heiko.schwarz@hhi.fraunhofer.de; thomas.schierl@hhi.fraunhofer.de;
detlev.marpe@hhi.fraunhofer.de

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_3
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00060

3.1 Introduction

The High Efficiency Video Coding (HEVC) standard is designed along the successful principle of block-based hybrid video coding. Following this principle, a picture is first partitioned into blocks and then each block is predicted by using either intra-picture or inter-picture prediction. While the former prediction method uses only decoded samples within the same picture as a reference, the latter uses displaced blocks of already decoded pictures as a reference. Since inter-picture prediction typically compensates for the motion of real-world objects between pictures of a video sequence, it is also referred to as motion-compensated prediction. While intra-picture prediction exploits the spatial redundancy between neighboring blocks inside a picture, motion-compensated prediction utilizes the large amount of temporal redundancy between pictures. In either case, the resulting prediction error, which is formed by taking the difference between the original block and its prediction, is transmitted using transform coding, which exploits the spatial redundancy inside a block and consists of a decorrelating linear transform, scalar quantization of the transform coefficients and entropy coding of the resulting transform coefficient levels.

Figure 3.1 shows a block diagram of a block-based hybrid video encoder with some characteristic ingredients of HEVC regarding its novel block partitioning concept. This innovative feature of HEVC along with its specific key elements will be one of the main subjects of this chapter. In a first step of this new block partitioning approach, each picture in HEVC is subdivided into disjunct square blocks of the same size, each of which serves as the root of a first block partitioning quadtree structure, the coding tree, and which are therefore referred to as coding tree blocks (CTBs). The CTBs can be further subdivided along the coding tree structure into coding blocks (CBs), which are the entities for which an encoder has to decide between intra-picture and motion-compensated prediction. The partitioning of pictures into CTBs and the partitioning of CTBs into CBs are described in Sects. 3.2.1 and 3.2.2, respectively. Both sections highlight the similarities and differences of the CTB/CB partitioning to the macroblock partitioning that is used in older video coding standards. The partitioning of CBs for the purpose of prediction is to a large degree independent of the coding tree structure and will be described in Sect. 3.2.3. Transform coding of the prediction residual at the CB level relies on the second, nested block partitioning quadtree structure, the so-called residual quadtree, and will be described along with its resulting transform blocks (TBs) in Sect. 3.2.4. Finally, Sect. 3.2.5 presents an experimental evaluation of some aspects of the block partitioning concept of HEVC and compares the HEVC design to that of older video coding standards.

In the second part of this chapter, the focus is on aspects of picture partitioning for the purposes of packetization and parallel processing. Section 3.3.1 describes the segmentation of a picture into slices, as already known from previous video coding standards, and the novel, optional fragmentation of a slice into slice segments. Aspects of high-level parallelization are discussed in Sect. 3.3.2 along with the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00061

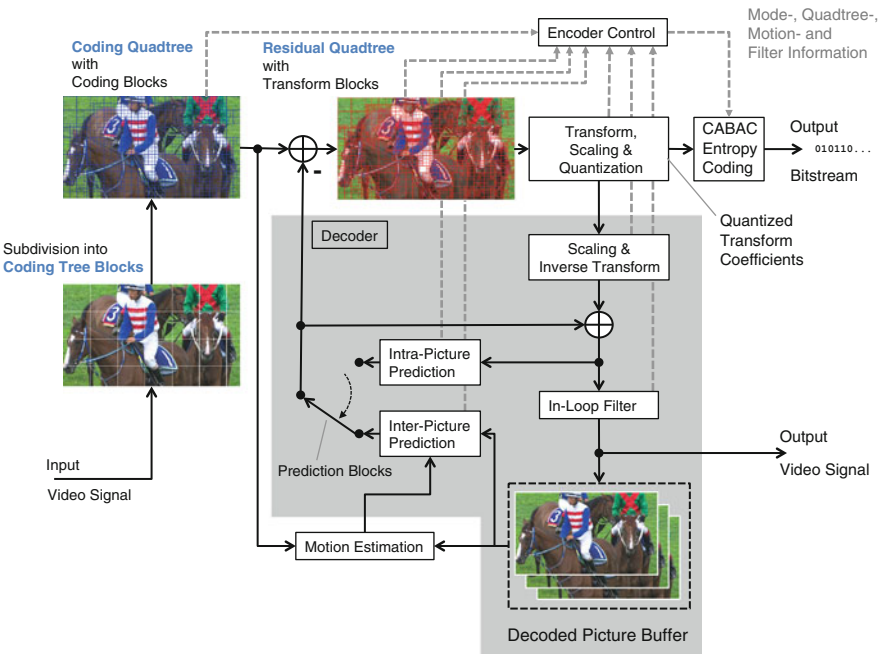


Fig. 3.1 Block diagram of an HEVC encoder with built-in decoder (*gray shaded*)

new features of tiles and wavefront parallel processing (WPP). Also, it is described how these picture partitioning features can be used together with the concept of slice segments for improved bitstream access. Section 3.3.3 deals with the same combination of tools but from the different perspective of support for ultra-low delay applications. Finally, in Sect. 3.3.4, the assets and drawbacks of tiles and WPP are discussed. The whole chapter is concluded in Sect. 3.4.

3.2 Block Partitioning for Prediction and Transform Coding

All ITU-T and ISO/IEC video coding standards since H.261 [14] follow the approach of block-based hybrid video coding, as it was already briefly discussed above and illustrated in Fig. 3.1. One significant difference between the different generations of video coding standards is that they provide different sets of coding modes for a block of samples. On the one hand, the selected coding mode determines whether the block of samples is predicted using intra-picture or inter-picture prediction. On the other hand, it can also determine the subdivision of a given block into subblocks used for prediction and/or transform coding. Blocks that are used for prediction are typically additionally associated with prediction parameters, such as motion vectors or intra prediction modes.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00062

In order to give the developers of encoder and decoder products as much freedom as possible while ensuring interoperability between devices of different manufacturers, the video coding standards only specify the bitstream syntax and the result of the decoding process,[1] while the encoding process is left out of scope. However, the coding efficiency of a particular encoder depends to a large extent on the encoding algorithm that is used for determining the values of the syntax elements written to the bitstream. This includes the selection of coding modes, associated prediction parameters, quantization parameters as well as quantization indices for the transform coefficients. A conceptually simple and very effective class of encoding algorithms are based on Lagrangian bit-allocation [31, 39, 43]. With these approaches, the used coding parameters p^* are determined by minimizing a weighted sum of the resulting distortion D and the associated number of bits R over the set $\mathscr{A}$ of available choices,

$$p^* = \arg \min_{\forall p \in \mathscr{A}} D(p) + \lambda \cdot R(p). \tag{3.1}$$

The Lagrange parameter λ is a constant that determines the trade-off between distortion D and the number of bits R and thus both the quality of the reconstructed video and the bit rate of the bitstream.

The coding efficiency that a hybrid video coding standard can achieve depends on several design aspects such as the used interpolation filters for sub-sample interpolation, the efficiency of the entropy coding, or the employed in-loop filtering techniques. However, the main source of improvement from one standard generation to the next is typically given by the increased number of supported possibilities for coding a picture or a block of samples. This includes, for example, an increased precision of motion vectors, a larger flexibility for choosing the coding order of pictures, an extended set of available reference pictures, an increased number of intra prediction modes, an increased number of motion vector predictors, an increased number of supported transform sizes as well as an increased number of block sizes for motion-compensated prediction.

In the following, we investigate the set of choices that are supported for partitioning a picture into blocks for motion-compensated prediction, intra-picture prediction, and transform coding. If we consider a given block of samples, different subdivisions into blocks used for prediction or transform coding are associated with different trade-offs between distortion and rate. When we subdivide a block into multiple subblocks and select the best prediction parameters for each subblock, we typically decrease the prediction error energy, but increase the bit rate required for transmitting the prediction parameters. Whether a subdivision is advantageous in rate-distortion sense depends on the actual block of samples. By extending the set of supported subdivision modes we typically increase the bit rate that is required for

[1]The standards specify an example decoding process. A decoder implementation is conforming to a standard if it produces the same output pictures as the specified decoding process. For older standards such as MPEG-2 Video, an accuracy requirement for the inverse transform is specified.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00063

signaling the selected modes, but decrease the resulting average rate-distortion cost for coding the prediction residuals, presuming a suitable encoder decision algorithm. One set $\mathscr{A}$ of partitioning modes improves the coding efficiency relative to another set if it yields a smaller expectation value,

$$\mathrm{E}\left\{\min_{\forall p \in \mathscr{A}} D(p) + \lambda \cdot R(p)\right\}, \tag{3.2}$$

of the Lagrangian rate-distortion cost for typical video content. It should, however, be noted that for a larger set of possible subdivision modes, in general, an encoder also requires a higher computational complexity for evaluating the set of supported modes. Hence, when designing a standard, a reasonable compromise between the potential coding efficiency and the required encoder complexity has to be chosen.

Due to continuous improvements in computing power, newer video coding standards support an increased set of coding options. In the development of HEVC, it was further taken into account that the coding of high and ultra-high definition video becomes more and more important. For dealing with such high resolutions, it is generally advantageous to support larger block sizes for both motion-compensated prediction and transform coding. But for adapting the block partitioning to the local properties of pictures, it is also important to additionally support small block sizes. Both objectives have been addressed in HEVC by introducing a hierarchical block partitioning concept based on a simple and unified but yet efficient quadtree syntax [27, 44]. In addition, this quadtree-based block partitioning concept allows the application of fast optimal tree pruning algorithms [5] in the encoder for determining the best block partitioning in terms of Lagrangian rate-distortion cost.

3.2.1 Coding Tree Blocks and Coding Tree Units

In all prior video coding standards [11, 12, 14–17] of the ITU-T and ISO/IEC, each picture of a video sequence is partitioned into so-called macroblocks. A macroblock consists of a 16×16 block of luma samples and, in the 4:2:0 chroma sampling format, two associated 8×8 blocks of chroma samples (one for each chroma component). The macroblocks can be considered as the basic processing units in these standards. For each macroblock of a picture, a coding mode has to be selected by the encoder. The chosen macroblock coding mode determines whether all samples of a macroblock are predicted using intra-picture prediction or motion-compensated prediction. Depending on the features supported in the actual standard, it can additionally determine the partitioning of the macroblock into subblocks that are used for motion-compensated prediction or intra-picture coding. The macroblock size of 16×16 luma samples represents the largest block size that can be used for signaling prediction parameters such as motion data.

Although video coding standards such as H.262 | MPEG-2 Video [16] and H.264 | MPEG-4 AVC [17] are used today for storing and transmitting

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00064

high-definition (HD) video content, with typical picture resolutions of 1280×720 or 1920 × 1080 luma samples, they have been primarily designed for video resolutions ranging from QCIF (176 × 144 luma samples) to standard definition (720 × 480 or 720×576 luma samples). Due to the popularity of HD video and the growing interest in Ultra HD (UHD) formats [13] with resolutions of, for example, 3840 × 2160 or even 7680 × 4320 luma samples, HEVC [18] has been designed with a focus on high resolution video. However, for such large picture resolutions, restricting the largest block size that can be used for signaling prediction parameters to 16 × 16 luma samples as in prior video coding standards is inefficient in rate-distortion sense [4, 26, 37]. For typical HD or UHD video content, many picture areas that can be described by the same motion parameters are much larger than blocks of 16 × 16 luma samples. Signaling a coding mode for each 16 × 16 macroblock would already require a substantial amount of the target bit rate. Furthermore, due to the increased spatial correlation between neighboring samples in high-resolution video, using transform sizes larger than 16 × 16 for coding the residual signal can also be advantageous for many image parts. The support of larger block sizes for intra-picture prediction, motion-compensated prediction and transform coding was one of the key aspects in many proposals for HEVC, for example [25, 27, 44].

Even though the coding of HD and UHD video was one important aspect in the HEVC development, the standard has been designed to provide an improved coding efficiency relative to its predecessor H.264 | MPEG-4 AVC for all existing video coding applications. While increasing the size of the largest supported block size is advantageous for high-resolution video, it may have a negative impact on coding efficiency for low-resolution video, in particular if low-complexity encoder implementations are used that are not capable of evaluating all supported sub-partitioning modes. For this reason, HEVC includes a flexible mechanism for partitioning video pictures into basic processing units of variable sizes.

As already mentioned, in HEVC, each picture is partitioned into square-shaped coding tree blocks (CTBs) such that the resulting number of CTBs is identical for both the luma and chroma picture components (assuming a non-monochrome video format).[2] Consequently, each CTB of luma samples together with its two corresponding CTBs of chroma samples and the syntax associated with these sample blocks is subsumed under a so-called coding tree unit (CTU). A CTU represents the basic processing unit in HEVC and is in that regard similar to the concept of a macroblock in prior video coding standards. The luma CTB covers a square picture area of $2^N \times 2^N$ luma samples. In the 4:2:0 chroma sampling format, each of the two chroma CTBs covers the corresponding area of $2^{N-1} \times 2^{N-1}$ chroma samples of one of the two chroma components. The parameter N is transmitted in the sequence parameter set and can be chosen by the encoder among the values $N = 4$, 5, and 6, corresponding to CTU sizes of 16 × 16, 32 × 32, and 64 × 64

[2]The profiles defined in version 1 of the HEVC standard [18] only support video in the 4:2:0 chroma sampling format; monochrome video is not supported in these profiles.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00065

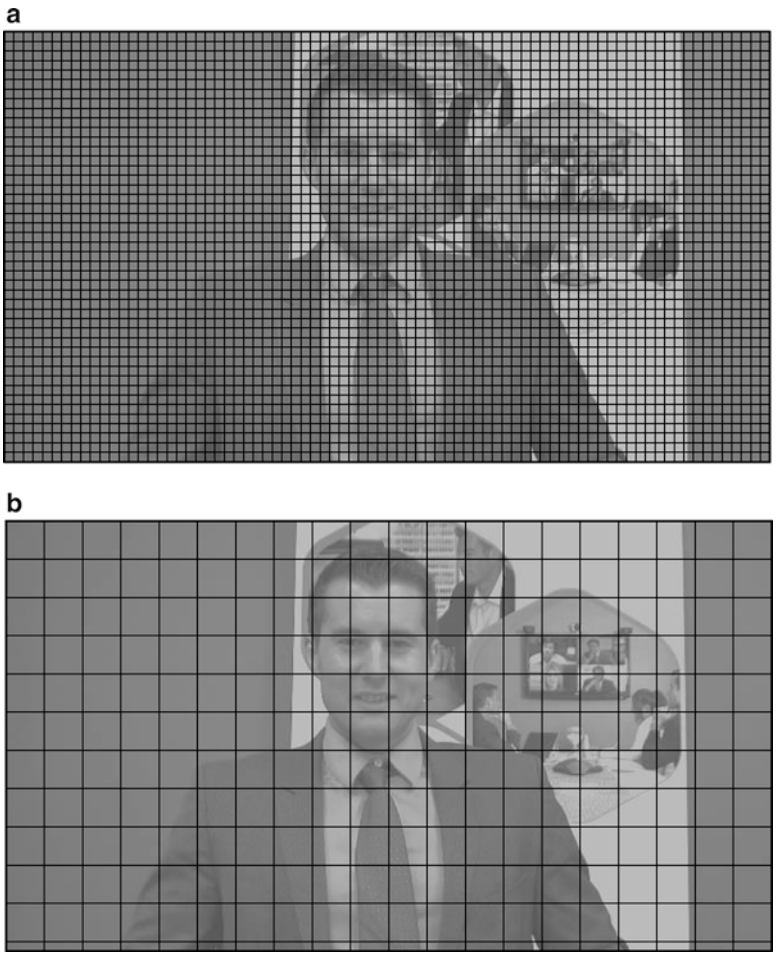


Fig. 3.2 Illustration of the partitioning of a picture with 1280 × 720 luma samples into macroblocks and coding tree units: (**a**) Partitioning of the picture into 16 × 16 macroblocks as found in all prior video coding standards of the ITU-T and ISO/IEC; (**b**) Partitioning of the picture into 64 × 64 coding tree units, the largest coding tree unit size supported in the Main profile of HEVC

luma samples. Larger CTU sizes[3] typically provide better coding efficiency, but may also increase the encoder/decoder delay, the memory requirements, and the computational complexity of the encoder process. The encoder has the freedom to choose the CTU size that provides the best trade-off for the targeted application.

For illustration, Fig. 3.2 shows the partitioning of a picture with 1280 × 720 luma samples into 16 × 16 macroblocks and 64 × 64 CTUs. It can be seen that a 16 × 16

[3]Here and in the following, the CTU size always refers to the corresponding luma CTB size.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00066

macroblock covers only a very small area of a picture, much smaller than the regions that can typically be described by the same motion parameters. Taking into account that some of the CTUs will be subdivided for assigning different prediction modes and parameters, as will be described in more detail in the following, the partitioning into 64 × 64 CTUs provides a more suitable description.

3.2.2 Coding Trees, Coding Blocks, and Coding Units

In the prior ITU-T and ISO/IEC video coding standards, a macroblock is not only used for partitioning a video picture; it also represents the processing unit for which a coding mode is chosen by the encoder. For each macroblock, it is decided whether all samples of the corresponding luma and chroma blocks are transmitted using inter-picture coding (i.e., motion-compensated prediction) or intra-picture coding. Furthermore, in both intra-picture and inter-picture coding, a macroblock can typically be subdivided into smaller blocks for the purpose of prediction and signaling of prediction parameters.

In the widely used High profile of H.264 | MPEG-4 AVC, three intra macroblock coding modes[4] are supported, which are referred to as Intra-4 × 4, Intra-8 × 8, and Intra-16×16. In the Intra-4×4 coding mode, the luma component of the macroblock is subdivided into 4 × 4 blocks. The samples of each 4 × 4 block are predicted based on the samples of already coded neighboring blocks and the prediction residual is coded using a 4 × 4 transform. When using the Intra-8 × 8 mode, the intra prediction and transform coding of the luma component is done for 8 × 8 blocks. For the Intra-16 × 16 mode, the entire 16 × 16 luma block is predicted using the samples of already coded neighboring macroblocks and the prediction residual is coded using a two-stage transform, which could be interpreted as a low-complexity variant of a 16 × 16 transform. In all three intra macroblock modes, the entire 8 × 8 chroma blocks are predicted and the residual is coded using a two-stage variant of an 8 × 8 transform. The partitioning of the luma component of a macroblock for intra-picture coding is illustrated in the bottom of Fig. 3.3.

For motion-compensated prediction, H.264 | MPEG-4 AVC supports four partitioning modes, which are referred to as Inter-16 × 16, Inter-16 × 8, Inter-8 × 16, and Inter-8 × 8 coding mode and are illustrated in the top of Fig. 3.3. In the Inter-16 × 16 mode, the motion-compensated prediction for all luma and chroma samples is done with the same set of motion parameters. For the Inter-16 × 8 and Inter-8 × 16 modes, the luma and chroma blocks of a macroblock are horizontally and vertically split into two rectangles of the same size, respectively, and each of the two resulting blocks is associated with a separate set of motion parameters. When the Inter-8 × 8 mode is used, the macroblock is split into four sub-macroblocks, each with a size of

[4] H.264 | MPEG-4 AVC additionally supports a so-called PCM mode, in which the macroblock samples are directly written into the bitstream.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00067

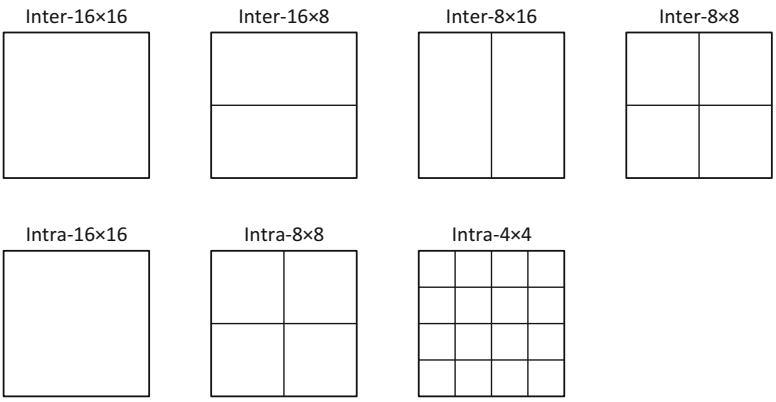


Fig. 3.3 Macroblock partitioning modes supported in the High profile of H.264 | MPEG-4 AVC for inter-picture coding (*top line*) and intra-picture coding (*bottom line*). If the Inter-8×8 is chosen, the 8 × 8 sub-macroblocks can be further partitioned into 8 × 4, 4 × 8, or 4 × 4 blocks

8 × 8 luma samples. Each of the resulting sub-macroblocks can be coded as a single 8 × 8 block, using a single set of motion parameters, or they can be further split into two 8 × 4, two 4 × 8, or four 4 × 4 blocks. Note that the supported subdivisions for 8×8 sub-macroblocks are the same as the subdivisions for macroblocks. For coding the luma prediction residual of inter-picture coded macroblocks, the High profile of H.264 | MPEG-4 AVC supports transform coding based on 4×4 and 8×8 blocks, the chosen transform size is signaled on a macroblock level. If a macroblock is coded in the Inter-8×8 mode and at least one of the sub-macroblocks is further subdivided, the syntax element signaling the transform size is not transmitted but the usage of the 4 × 4 transform is inferred; transform coding across boundaries of blocks used for motion-compensated prediction is not supported. The chroma residual signals in inter-picture coding modes are always coded using the 4 × 4 transform.

In HEVC, the basic processing units into which video pictures are partitioned can be as large as 64 × 64 luma samples. A direct application of the H.264 | MPEG-4 AVC macroblock syntax to the coding tree units in HEVC would cause some problems. On the one hand, choosing between intra-picture and motion-compensated prediction for large blocks is unfavorable in rate-distortion sense. In P and B slices, typically most of the samples can be well predicted using motion-compensated prediction. Only for a small amount of samples, intra-picture coding is advantageous in rate-distortion sense. If the standard would allow to choose between motion-compensated prediction and intra-picture coding only on the level of coding tree units, it would result in a significant loss in coding efficiency. Actually, it has been shown [21, 35] that enabling the decision between intra-picture and inter-picture coding at units smaller than a 16 × 16 macroblock can increase the coding efficiency. On the other hand, for allowing such fine block structures for motion-compensated prediction as in H.264 | MPEG-4 AVC, the concept of macroblock and sub-macroblock modes would have to be extended over additional hierarchy levels

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00068

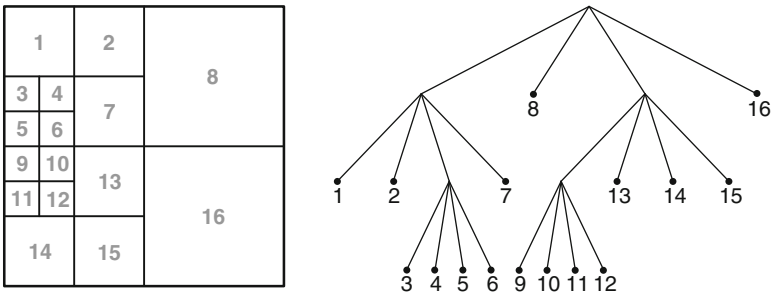


Fig. 3.4 Example for the partitioning of a 64 × 64 coding tree unit (CTU) into coding units (CUs) of 8 × 8 to 32 × 32 luma samples. The partitioning can be described by a quadtree, also referred to as coding tree, which is shown on the *right*. The numbers indicate the coding order of the CUs

yielding a complicated syntax. Additionally, if the transform size cannot change within a coding tree block, the encoder cannot well adapt to the local statistics in a video picture.

In order to overcome these potential issues, another processing unit, called coding unit (CU), has been introduced in HEVC. As illustrated for an example in Fig. 3.4, a CTU can be split into multiple coding units (CUs) of variable sizes. For that purpose, each CTU contains a quadtree syntax, also referred to as coding tree, which specifies its subdivision into CUs. Similarly as a CTU, a CU consists of a square block of luma samples, the two corresponding blocks of chroma samples (for non-monochrome video formats), and the syntax associated with these sample blocks. The luma and chroma sample arrays that are contained in a CU are referred to as coding blocks (CB). The subdivision of the chroma CTBs of a CTU is always aligned with that of the luma CTB. Thus, in the 4:2:0 chroma sampling format, each $2^N \times 2^N$ luma CB is associated with two $2^{N-1} \times 2^{N-1}$ chroma CBs.

At the CTU level, a flag named `split_cu_flag` is included into the bitstream, which indicates whether the complete CTU forms a CU or whether it is split into four equally-sized blocks corresponding to square luma sample blocks. If the CTU is split, for each of the resulting blocks, another `split_cu_flag` is transmitted specifying whether the block represents a CU or whether it is further split into four equally-sized blocks. This hierarchical subdivision is continued until none of the resulting blocks is further subdivided. The minimum size of CUs is signaled in the sequence parameter set, it can range from 8 × 8 luma samples to the size of the CTU, inclusive. When the minimum CU size is reached in the hierarchical subdivision process, no splitting flags are transmitted for the corresponding blocks; instead it is inferred that these blocks are not further split. Hence, if a low-complexity encoder is used that does never use coding blocks smaller than a particular size, the CU size in the sequence parameter can be set accordingly, which avoids the transmission of unnecessary splitting flags. In typical encoder settings, the maximum range of supported CU sizes is exploited so that CUs ranging from 8 × 8 to 64 × 64 luma samples can be used.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00069

The CUs inside a CTU are coded in a depth-first order. This coding order is also referred to as z-scan and is illustrated in Fig. 3.4. It ensures that for each CU, except those located at the top or left boundary of a slice, all samples above the CU and left to the CU have already been coded, so that the corresponding samples can be used for intra prediction and the associated coding parameters can be used for predicting the coding parameters of the current CU.

The horizontal and vertical size of a video picture, in luma samples, has to be an integer multiple of the minimum CU size,[5] in luma samples, transmitted in the sequence parameter set, but it does not need to be an integer multiple of the CTU size. If the horizontal or vertical size of the video pictures does not represent an integer multiple of the CTU size (as it is, for example, the case in Fig. 3.2b), the CTUs at the borders are inferred to be split until the boundaries of the resulting blocks coincide with the picture boundary. For this enforced splitting no splitting flags are transmitted, but the resulting blocks can be further split using the quadtree syntax described above. The CUs that lie outside the picture area not coded.

The CUs represent the processing units to which a coding mode is assigned. For each CU, it is decided whether the luma and chroma samples are predicted using intra-picture prediction or motion-compensated prediction. In that respect, CUs in HEVC are similar to macroblocks in older video coding standards. However, in contrast to macroblocks, CUs have variable sizes. Multiple CUs form a CTU, which represents the basic processing unit used for partitioning a picture. For intra-picture prediction, motion-compensated prediction, and transform coding of the prediction residuals, a CU can be further split into smaller blocks along the coding tree structure as will be discussed in the following subsections. The main advantage of introducing the coding tree structure is that in this way an elegant and unified syntax is obtained for specifying the partitioning of CTUs into blocks that are used for intra-picture prediction, motion-compensated prediction, and transform coding.

3.2.3 Prediction Blocks and Prediction Units

For each CU, a prediction mode is signaled inside the bitstream. The prediction mode indicates whether the CU is coded using intra-picture prediction or motion-compensated prediction. If intra-picture prediction is chosen, one of the 35 supported spatial intra prediction modes has to be selected for the luma CB and signaled inside the bitstream. If the CU has the minimum CU size specified in the sequence parameter set, the luma CB can also be decomposed into four equally-sized square subblocks, in which case a separate intra prediction mode is transmitted for each of these subblocks. Independent of the CU size, a single chroma intra

[5] Videos with picture sizes that do not represent an integer multiple of the minimum CU size can be coded by extending the picture area using arbitrary sample values and specifying a conformance cropping window in the sequence parameter set.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00070

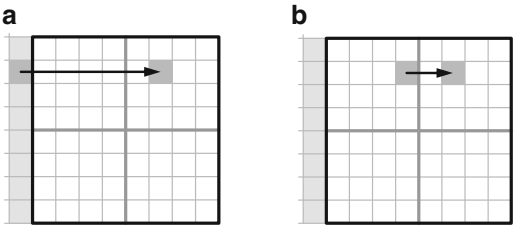


Fig. 3.5 Illustration of the horizontal intra prediction of a selected sample inside an 8×8 coding block with 4×4 transform blocks, if the intra prediction is applied on the basis of coding blocks (**a**) or transform blocks (**b**)

prediction mode is selected for a CU and signaled inside the bitstream. The chroma intra prediction mode applies to both chroma CBs and can be selected among five candidates, where one of the candidates represents the intra prediction mode chosen for the luma CB or the first luma intra block in case four intra prediction modes are transmitted for the luma CB. The reason why the signaling of four luma intra prediction modes is only supported for the minimum CU size is the following. If we consider a picture block of $2^N \times 2^N$ luma samples that is larger than the minimum CU size, then partitioning the block into four CUs and transmitting a luma intra prediction mode for each of the CUs is, for the luma component, the same as coding the entire block as one CU and transmitting an intra prediction mode for the four luma subblocks. Even though these partitioning methods do not yield the same result for the chroma components, they can be seen as redundant syntax features. And since the impact of coding the chroma components on the overall coding efficiency is usually very small, the corresponding redundant syntax is avoided and a subdivision of the luma CB for signaling intra prediction modes is only supported for the minimum CU size.

The actual intra prediction is not always applied to the blocks for which the intra prediction modes are signaled. A coding block can be split into multiple transform blocks, which represent the units to which a single two-dimensional transform is applied for coding the prediction residuals. As will be discussed in Sect. 3.2.4, the subdivision of the coding blocks of a CU into transform blocks is specified by a second quadtree structure, for which the CU represents the quadtree root. If a luma CB is subdivided into four subblocks for signaling the intra prediction modes, it is also subdivided for the purpose of transform coding so that all samples inside a transform block are always predicted using the same intra prediction mode. It is, however, possible that a block for which a single intra prediction mode is transmitted is further partitioned into multiple transform blocks. Since the correlation between two image samples decreases with the distance between the samples (for any given direction), on average, a better prediction signal is obtained if we use reconstructed samples that are closer to the samples we want to predict. Due to this reason, the intra prediction is done on the basis of transform blocks. The effect is illustrated in Fig. 3.5 for the example of horizontal intra prediction of an 8×8 coding block that is split into 4×4 transform blocks. It should be noted that, on the one hand side, the efficiency of intra prediction decreases with increasing the size of the transform blocks, since the average distance between a predicted sample and the reference samples used for prediction increases. On the other hand side, however,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00071

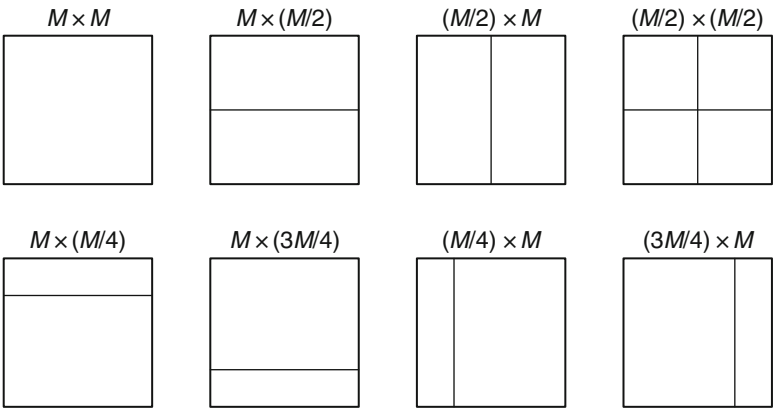


Fig. 3.6 Supported partitioning modes for splitting a coding unit (CU) into one, two, or four prediction units (PU). The $(M/2) \times (M/2)$ mode and the modes shown in the *bottom row* are not supported for all CU sizes

the average coding efficiency of transform coding (in terms of the average mean squared error for a given bit rate) typically increases with the transform size [1, 42]. Hence, the syntax feature that allows the splitting of a coding block (or a block used for signaling the intra prediction mode) into multiple transform blocks provides the possibility to select a suitable trade-off between the intra prediction and transform coding efficiency for the considered block. From a different point of view, it can also be argued that coding a single intra prediction mode for multiple transform blocks (i.e., the blocks that are actually used for intra prediction) represents a way for reducing the bit rate required for transmitting the intra prediction modes.

If a CU is coded using inter-picture prediction, the luma and chroma CBs can be further split into so-called prediction blocks. A prediction block (PB) is a block of samples of the luma or a chroma component that uses the same motion parameters for motion-compensated prediction. The motion parameters include the number of motion hypotheses (which is either one or two) as well as the reference picture index and motion vector for each of the motion hypotheses. For both chroma CBs of a CU, the same splitting as for the luma CB is used. The luma PB and chroma PBs, together with the associated syntax, form a prediction unit (PU). For each PU, a single set of motion parameters is signaled in the bitstream, which is used for motion-compensated prediction of the luma PB and the chroma PBs.

HEVC supports eight different modes for partitioning a CU into PUs. As illustrated in Fig. 3.6, a CU can either be coded as a single PU or it can be split into two or four rectangular PUs. The partitioning mode in which the entire CU is coded as a single PU is referred to as $M \times M$ mode. If a CU is split into four PUs, the resulting PUs represent square blocks of the same size and the partitioning mode is referred to as $(M/2) \times (M/2)$ mode. Since the splitting of a CU into four equally-sized square PUs is conceptually equivalent to splitting the corresponding picture block into four CUs and coding each of these CUs as

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00072

a single PU, the $(M/2) \times (M/2)$ mode is only supported for the minimum CU size that is signaled in the sequence parameter set. For splitting a CU into two PUs, HEVC supports six partitioning modes. In the $M \times (M/2)$ mode, the CU is vertically subdivided into two rectangular PUs of the same size. Similarly, the $(M/2) \times M$ mode horizontally subdivides the CU in two PUs of the same size. In addition to these symmetric partitioning modes, four asymmetric partitioning modes are supported, which subdivide the CU into two rectangular PUs of different sizes, as is illustrated in the bottom row of Fig. 3.6. One of the resulting PUs has a rectangular shape with one side having a length equal to the width and height M of the CU and the other side having a length equal to $M/4$; the other PU covers the remaining rectangular area of the CU. The asymmetric partitioning modes are only supported for CU sizes larger than 8×8 luma samples. Furthermore, for minimizing the worst-case memory bandwidth, the $(M/2) \times (M/2)$ mode is only supported if the selected minimum CU size is larger than 8×8 luma samples, so that blocks of 8×4 and 4×8 are the smallest block sizes that can be used for motion-compensated prediction. In addition, PUs of 8×4 and 4×8 luma samples are restricted to use a single motion hypothesis.

Supporting more modes for partitioning a picture block into subblocks used for motion-compensated prediction typically provides the potential for increasing the coding efficiency. However, this potential can only be exploited if an encoder evaluates a significant number of the supported partitioning modes. Otherwise, the syntax overhead associated with an increased number of modes may actually decrease the coding efficiency in comparison to supporting a smaller set of modes. The set of partitioning modes supported in HEVC has been selected as a reasonable trade-off. In addition, HEVC provides the possibility to disable the asymmetric partitioning modes via a syntax element coded in the sequence parameter set (SPS). If these modes are disabled, the entropy coding of the partitioning mode is modified so that the remaining partitioning modes can be signaled with less bits (see Chap. 8). This feature is particularly useful for low-complexity encoders that do not have the computational resources to evaluate all possible partitioning modes.

In the development of video coding standards, one key aspect for improving the coding efficiency from one generation of standards to the next was to increase the number of supported block sizes for motion-compensated prediction. If we concentrate on profiles intended for the coding of progressive video, H.262 | MPEG-2 Video supports only a single block size of 16×16 luma samples for specifying motion parameters. In H.263 and MPEG-4 Visual, a subdivision of the 16×16 macroblocks into four 8×8 blocks is additionally allowed. The concept of subdividing a 16×16 macroblock into smaller blocks is further extended in H.264 | MPEG-4 AVC, where block sizes from 4×4 to 16×16 luma samples, including non-square blocks, are supported. Now in HEVC, the concept is additionally extended towards larger block sizes, resulting in motion compensation block sizes that range from 4×8 and 8×4 luma samples to 64×64 luma samples. The motion compensation block sizes that are supported in different standards (excluding special tools for interlaced video) are summarized in Table 3.1.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00073

Table 3.1 Comparison of motion compensation block sizes supported in different standards

Video coding standard	Supported block sizes for motion-compensated prediction
H.262 | MPEG-2 Video	16 × 16
H.263	16 × 16, 8 × 8
MPEG-4 Visual	16 × 16, 8 × 8
H.264 | MPEG-4 AVC	16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8, 4 × 4
HEVC	64 × 64, 64 × 48, 64 × 32, 64 × 16, 48 × 64, 32 × 64, 16 × 64, 32 × 32, 32 × 24, 32 × 16, 32 × 8, 24 × 32, 16 × 32, 8 × 32, 16 × 16, 16 × 12, 16 × 8, 16 × 4, 12 × 16, 8 × 16, 4 × 16, 8 × 8, 8 × 4, 4 × 8

3.2.4 Residual Quadtree Transform, Transform Blocks, and Transform Units

As already mentioned above, for transform coding of the prediction residuals in HEVC, a CB can be partitioned into multiple transform blocks (TBs). A transform block represents a square block of samples of a color component on which the same two-dimensional transform is applied for coding the residual signal. The partitioning of a luma CB into luma TBs is carried out recursively based on a quadtree approach. The corresponding structure, denoted as the residual quadtree (RQT), determines for each luma CB at the root of the RQT a collection of luma TBs at the leaves of the RQT in such a way that the union of corresponding disjoint luma TBs is covering the whole associated luma CB. Figure 3.7 shows an example of a 64 × 64 luma CTB that is subdivided recursively into luma CBs and luma TBs along the corresponding nested quadtree structures of coding tree and residual quadtrees. In general, the partitioning of chroma CBs into chroma TBs is described by the same residual quadtree. As will be described below, there is, however, one exception, for which the splittings of the luma and chroma CBs of the same CU are not the same.

By allowing different transform block sizes, the residual quadtree transform enables the adaptation of the transform basis functions to the varying space-frequency characteristics of the residual signal. Larger transform block sizes, which have larger spatial support, provide better frequency resolution. However, smaller transform block sizes, which have smaller spatial support, provide better spatial resolution. The trade-off between the two, spatial and frequency resolution, can be freely chosen by the encoder control, for example, based on Lagrangian optimization techniques. In the next subsection, the RQT structure is described in detail followed by the description of its related parameter signaling scheme as well as a brief discussion of a fast encoder implementation for RQT using an early-pruning criterion.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00074

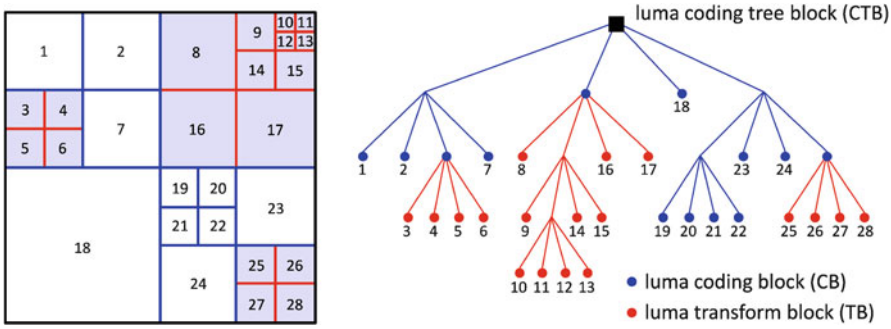


Fig. 3.7 Example for the partitioning of a 64 × 64 luma coding tree block (*black*) into coding blocks (*blue*) and transform blocks (*red*). In the illustration on the *right*, the *blue lines* show the corresponding coding tree with the coding tree block (*black square*) at its root and the coding blocks (*blue circles*) at its leaf nodes; the *red lines* show the non-degenerated residual quadtrees with the transform blocks (*red circles*) as leaf nodes. Note that the transform blocks chosen identical to the corresponding coding blocks are not explicitly marked in this figure. The numbers indicate the coding order of the transform blocks

3.2.4.1 Residual Quadtree Structure

Each RQT is restricted by three parameters: The maximum depth $d_{\max}$ of the tree, the minimum allowed transform size $n_{\min}$ and the maximum allowed transform size $n_{\max}$, where the latter two are given as the binary logarithm of the transform block width. Both $n_{\min}$ and $n_{\max}$ can vary within the range from two to five, which means that transform block sizes from 4×4 to 32×32 samples are supported in HEVC. The maximum allowed depth $d_{\max}$ of the RQT restricts the number of subdivision levels. For example, a value of $d_{\max} = 1$ means that a luma CB can either be coded as a single TB or it can be split into 4 TBs, but no further splitting is allowed.

As a result of the interaction of these limits on transform block size and tree depth, there are conditions that imply an internal (i.e., branching) node or a leaf (i.e., non-branching) node in the RQT. As an example, consider a case, in which the luma CB of the CU associated with the RQT root has a size of 64 × 64 samples, the maximum depth is $d_{\max} = 0$ and the maximum transform size is equal to 32 × 32, i.e., $n_{\max} = 5$. In HEVC, if the luma CB size is larger than the maximum transform size, as in the mentioned example, the luma CB is forced to be subdivided to comply with the limitations on the transform size.

If the decorrelating transform is applied across multiple prediction blocks, the transformed residual signal often includes block edges at the corresponding prediction boundaries, which increase the energy in the high-frequency transform coefficients and, hence, decrease the coding efficiency. Due to this reason, HEVC includes a special condition for implicit subdivision when the maximum RQT depth is equal to zero. If $d_{\max} = 0$, the CU is predicted using motion compensation, and the associated prediction partitioning consists of more than one PU, the luma CB is always split into four TBs [32]. For $d_{\max} > 0$ and the case of inter-predicted

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00075

CUs, the RQT subdivision is independent of the PU partitioning, which may lead to a transform block covering several different PBs having the same CB as the root. Note that even though applying the transform across PB boundaries can cause unwanted coding efficiency degradations for several CBs, it may also increase the coding efficiency for other CBs. Experimental results [23] showed that giving the encoder the freedom to decide whether a transform is applied across PB boundaries increases the average coding efficiency by about 0.4–0.7 % Bjøntegaard Delta bit rate (BD rate) [3].

As discussed in Sect. 3.2.3, for intra-predicted CUs, the splitting of the luma and chroma CBs into transform blocks does not only determine the size of the transform that is applied for residual coding, but also the size of the blocks for which a single intra prediction signal is generated. If the size of a luma CB size is equal to the minimum CB size signaled in the sequence parameter set, it is possible to signal four luma intra prediction modes for the luma CB, each for one of the four equally-sized square subblocks. In that case, a transform block cannot span the complete luma CB, since the neighboring subblocks within the luma CB that precede a current subblock in coding order have to be fully reconstructed for generating the prediction signal for the current subblock. As a consequence, if a luma CB is subdivided into four subblocks for transmitting intra prediction modes, it is also subdivided into four TBs. Note, however, that each of the resulting TBs may then be further subdivided.

A leaf node is implied in the RQT when the minimum transform size is reached, i.e., if the binary logarithm n of the actual transform block size satisfies the condition $n = n_{\min}$. However, the recursive subdivision is also restricted by the maximum RQT depth, even when smaller TBs are allowed by the minimum transform size, which corresponds to the case that the actual RQT depth d conforms with $d = d_{\max}$ and that the above-mentioned conditions on implicit subdivision are not fulfilled.

Note that at most one single transform tree syntax is transmitted for each CU, and thus, the same RQT structure determines the TBs of all color components, resulting in the same subdivision of luma and chroma components. An exception to this rule, however, is given for the case of a 4×4 luma TB in a 4:2:0 chroma-formatted video signal, where the subdivision would lead to a corresponding 2×2 chroma TB, which is not supported in HEVC. Therefore, an RQT is allowed to split an 8×8 luma TB, but not the corresponding 4×4 chroma TB, leading to a different interpretation of the corresponding RQT structure for luma and chroma in this particular case. A transform unit (TU) represents a luma TB greater than 4×4 samples or four luma TBs with a size of 4×4 samples, the corresponding two chroma TBs, and the associated syntax structures.

3.2.4.2 Parameter Signaling

The RQT parameters introduced in the previous subsection, i.e., the maximum RQT depth $d_{\max}$ and the binary logarithms of the minimum and maximum transform sizes $n_{\min}$ and $n_{\max}$ are transmitted in the bitstream at the SPS level. Regarding the RQT depth, different values can be specified and signaled for intra- and inter-predicted

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00076

CUs. Furthermore, it should be kept in mind that the transmitted depth values do not necessarily correspond to the number of subdivisions when subdivisions are forced, due to resulting TB sizes greater than the maximum transform size, as already mentioned above.

Since the RQT structure is nested in the coding tree, it has to be signaled for each coding tree leaf node, i.e., for each CU after transmission of CU prediction mode, PU partitioning, and PU related syntax, provided that the residual signal of the CU is represented by one or more non-zero transform coefficients levels. In that case, the syntax element `split_transform_flag` is transmitted for every RQT node to indicate whether it is a leaf node (with a value of 0) or an internal node (with a value of 1). Note that for the cases presented in the previous subsection, where a signaling of this flag would be redundant, `split_transform_flag` is not explicitly signaled but inferred at the decoder side instead.

In addition to the actual structure of the RQT but often also as its replacement, it is signaled whether there are significant, i.e., non-zero transform coefficient levels present in a particular TB or the whole CU. For CUs using motion-compensated prediction, one single coded block flag (cbf) signals the information whether at least one non-zero transform coefficient level is transmitted for the whole CU. When this so-called `rqt_root_cbf` is equal to 1, the RQT structure is signaled as described above. Otherwise, no further residual information is transmitted and all transform coefficient levels are inferred to be equal to zero. In the latter case, all residual sample values are also equal to zero and no RQT syntax is transmitted. The syntax element `rqt_root_cbf` is especially useful for coding of video signals at low bit rates or for coding picture areas that can be motion-compensated predicted in a sufficiently accurate way, because, with one single flag, a potentially large number of transform coefficient levels equal to zero can be signaled very efficiently. Also note that in the case of a skipped CU, i.e., a motion-compensated CU with `cu_skip_flag` set equal to 1, no residual and hence, no transform syntax including the `rqt_root_cbf` is transmitted. For intra-predicted CUs, however, the `rqt_root_cbf` is always inferred to be equal to 1, since in that case, a reasonable assumption is that at least some of the residual transform coefficient levels are not equal to zero.

Furthermore, in case of `rqt_root_cbf` $= 1$, an additional cbf is transmitted for each luma TB and each of the two associated chroma TBs. The significance for luma TBs is signaled at the RQT leaf node level using the syntax element `cbf_luma`, while for the chroma components, the flags `cbf_cb` and `cbf_cr` are coded interleaved with the `split_transform_flag` symbols, i.e., at an internal RQT node. This concept enables an efficient signaling for square blocks in the video signal for which one or both of the chroma residual signals are equal to zero, but the luma residual signal is not equal to zero. Under certain conditions, the signaling of cbfs for luma and chroma TBs is also redundant and can be inferred instead. Details on this redundancy reduction can be found in [24, 30].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00077

3.2.4.3 Fast Encoder Control

The number of different RQT partitionings as well as the number of different coding tree partitionings grows faster than the double exponential expression $2^{4^{d-1}}$ with increasing depth d of the tree. However, as further explained in [25], by application of the generalized BFOS algorithm [5], the derivation of an optimal partitioning in a rate-distortion sense can be achieved without the need of a brute-force exhaustive search that requires to compare each partitioning option with all of its competing options. In fact, it can be shown that without applying any early termination strategy, the computational complexity of the fast tree-pruning process is proportional to $(4^d - 1)/3$, which is the number of internal nodes of a quadtree with maximum depth d. However, in order to further reduce the computational complexity for the tree-growing process at the encoder side, heuristic early-pruning techniques [25,38] can be additionally applied.

In case of the RQT partitioning, such an algorithm was presented in [38]. Here, the idea is that the evaluation of further subdivisions at a given RQT node should be terminated when all magnitudes of unquantized transform coefficients are below an appropriately chosen, quantizer step size-dependent threshold. It was shown that by applying such a strategy, the encoder runtime can be reduced by about 5–15 % with only minor impact on coding efficiency. Also, the reduction of encoder runtime is consistently higher for a larger maximum RQT depth, since with a larger search space for the optimal RQT partitioning, typically larger improvements in complexity reduction can be achieved if only a subset of the whole search space is considered. For more details, the reader is referred to [38].

3.2.5 Performance

For evaluating selected design aspects of the HEVC block structures, we performed coding experiments for two different application scenarios. The first scenario considers the coding of high-resolution video with entertainment quality, while the second scenario addresses interactive video applications such as video conferencing.

For the scenario of entertainment applications, we selected five HD sequences of varying content, all having a resolution of 1920×1080 luma samples. The sequences were coded using a dyadic high-delay hierarchical prediction structure with groups of eight pictures [36] and four active reference pictures. Random access points, which are coded as I pictures, are inserted in regular intervals of about 1 s. In order to enable clean random access, pictures that follow an I picture in both coding and display order are restricted in a way that they do not reference any pictures that precede the I picture in coding or display order. With exception of the random access pictures, all pictures are coded as B pictures. The quantization parameter (QP) is increased by 1 from one hierarchy level to the next and the QP for the B pictures of the lowest hierarchy level is increased by 1 relative to that of the random access I pictures. All pictures are coded as a single slice.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00078

Table 3.2 Summary of the test sequences for the two considered application scenarios that are used in the comparisons. All sequences are given in the 4:2:0 chroma sampling format with a bit depth of 8 bit per sample and have a duration of 10 s. The resolution is specified in luma samples

Entertainment applications			Interactive applications		
Sequence name	Resolution	Frame rate	Sequence name	Resolution	Frame rate
Kimono	1920 × 1080	24 Hz	Four People	1280 × 720	60 Hz
Park Scene	1920 × 1080	24 Hz	Johnny	1280 × 720	60 Hz
Cactus	1920 × 1080	50 Hz	Kristen & Sara	1280 × 720	60 Hz
BQ Terrace	1920 × 1080	60 Hz	Vidyo 1	1280 × 720	60 Hz
Basketball Drive	1920 × 1080	50 Hz	Vidyo 2	1280 × 720	60 Hz
			Vidyo 3	1280 × 720	60 Hz

The hierarchical prediction structure chosen for the entertainment application scenario provides a very high coding efficiency, but it is associated with a structural delay of eight pictures. For interactive video applications, such a high delay is not acceptable. Here, we used a dyadic low-delay hierarchical prediction structure with groups of four pictures [36] and four active reference pictures. For this coding structure, all pictures are coded in display order, but the QP is varied depending on the hierarchy level. Similarly as for the coding structure chosen for the entertainment scenario, the QP is increased by 1 from one hierarchy level to the next. Except the first picture of a video sequence, which is coded as I picture, all pictures are coded as B pictures. The QP for the I picture is decreased by 1 relative to the QP for the B pictures of the lowest hierarchy level. Furthermore, all pictures are coded as a single slice. For the application scenario of interactive video application, we selected six sequences with typical video conferencing content. All of these sequences have a resolution of 1280 × 720 luma samples.

The chosen test sequences for both scenarios are summarized in Table 3.2. They represent a subset of the sequences that have been used by the standardization group during the development of HEVC. Furthermore, the configurations for both selected application scenarios are consistent with the test conditions [19] recommended by the standardization group. All coding experiments were performed with the HEVC reference software, version HM10.1 [20]. The encoder employs the Lagrangian bit allocation technique described in [31]. For evaluating the impact of particular aspects of the HEVC block structures and comparing it to the block structures supported in older standards, selected block sizes for prediction and transform coding were disabled in some of the simulations. This was partly done by specifying corresponding parameters, such as the CTU size, the minimum CU size, or the minimum and maximum transform size, in the encoder configuration files. Other features were disabled by modifying the corresponding source code. In order to compare the HEVC design to that of H.264 | MPEG-4 AVC, we additionally enabled 4 × 4 PUs in some experimental settings by slightly modifying the HEVC syntax.

The efficiency of different configurations was compared by encoding each of the test sequences for an application scenario at ten different quantization settings. Therefore, the QP for I pictures was varied from 20 to 38, inclusive, in steps of 2.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00079

As quality measure, we used a weighted sum of the average PSNR values for the luma and chroma component, defined by

$$\mathrm{PSNR}_{\mathrm{YUV}} = (6 \cdot \mathrm{PSNR}_{\mathrm{Y}} + \mathrm{PSNR}_{\mathrm{U}} + \mathrm{PSNR}_{\mathrm{V}})/8, \tag{3.3}$$

where $\mathrm{PSNR}_{\mathrm{Y}}$, $\mathrm{PSNR}_{\mathrm{U}}$, and $\mathrm{PSNR}_{\mathrm{V}}$ represent the average PSNR values, averaged over the pictures, for the individual color component. For video with a bit depth of 8 bit per sample, the picture PSNR for a particular color component is given by

$$\mathrm{PSNR} = -10 \cdot \log_{10}\left(\frac{1}{255^2 \cdot W \cdot H}\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\left(s(x,y) - s'(x,y)\right)^2\right), \tag{3.4}$$

where s and s' represent the $W \times H$ sample arrays of the original and reconstructed color component, respectively. By measuring the reconstruction quality using the $\mathrm{PSNR}_{\mathrm{YUV}}$ and the average bit rate of the generated bitstreams for ten different quantization settings, we obtained a rate-distortion curve for each configuration and sequence, which characterizes the coding efficiency. In order to express the difference in coding efficiency between two configurations as a single number, we calculated an average bit-rate saving. Therefore, we interpolated the rate-distortion curves of the two considered configurations for a test sequence in the logarithmic domain using cubic splines with the "not-a-knot" condition at the border points and determined the average bit-rate saving for the sequence by numerical integration with 1,000 equal-sized subintervals.[6] Finally, the average bit-rate savings for the application scenarios given in the following were obtained by averaging the bit-rate savings over the test sequences.

In a first experiment, we investigated the impact of supporting different block sizes for signaling motion parameters. In order to exclude the effect of different transform sizes, the maximum TU size was set to 4×4 for all investigated configurations. As reference configuration, we used a setting in which only 16×16 PUs are enabled, similarly as in H.262 | MPEG-2 Video. Based on this reference setting, we successively enabled additional PU sizes as summarized in Table 3.3. It can be seen that also for high-resolution video, supporting PU sizes smaller than 16×16, as has been done in developing H.263, MPEG-4 Visual, and H.264 | MPEG-4 AVC, improves the coding efficiency. However, the biggest gain is achieved by additionally supporting PU sizes larger than 16×16 luma samples. While most of the coding gain relative to the reference configuration can already be obtained by enabling all square PUs from 4×4 to 64×64 luma samples, the additional support of rectangular PUs with symmetric and asymmetric CU partitioning modes further improves the coding efficiency. In order to include design

[6]The employed approach for calculating an average bit-rate saving between two rate-distortion curves represents a generalization, that is also applicable to experimental data with more than four rate-distortion points, of the often used Bjøntegaard Delta bit rate (BD-rate) [3].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00080

Table 3.3 Coding efficiency improvement for successively enabling PUs of selected sizes. The shown average bit-rate savings are measured relative to a configuration with 16 × 16 PUs only. In all configurations, the transform coding is done using a 4 × 4 transform

Enabled PU sizes	Entertainment applications	Interactive applications
16 × 16 and 8 × 8 PUs	2.7 %	4.4 %
Square PUs from 4 × 4 to 16 × 16	6.1 %	5.6 %
Square PUs from 4 × 4 to 32 × 32	15.4 %	23.0 %
Square PUs from 4 × 4 to 64 × 64	18.7 %	30.3 %
All modes except asym. (+4 × 4 PUs)	20.0 %	31.0 %
All HEVC PU sizes (+4 × 4 PUs)	20.7 %	33.0 %

Table 3.4 Coding efficiency improvement for successively increasing the maximum TU size. The shown average bit-rate savings are measured relative to a configuration with 4 × 4 TUs only. In all configurations, all supported CU and PU sizes are enabled and the minimum TU size is 4 × 4

Maximum enabled TU size	Entertainment applications	Interactive applications
Maximum TU size of 8 × 8	6.8 %	8.5 %
Maximum TU size of 16 × 16	11.9 %	14.7 %
Maximum TU size of 32 × 32	13.9 %	17.5 %

aspects of H.264 | MPEG-4 AVC in the comparison, we enabled 4 × 4 PUs in most of the configurations, even though they are not supported in the HEVC. This small syntax modification does not change the interpretation of the results, since the average bit-rate saving for additionally enabling 4 × 4 PUs in HEVC is typically less than 0.1 % for high-resolution video [22].

The second experiment evaluates the impact of supporting different transform sizes. Here, we enabled all CU and PU sizes supported in the HEVC standard, but restricted the maximum TU size. In the reference configuration, only 4 × 4 TUs were allowed, similar as in the Main profile of H.264 | MPEG-4 AVC. While the minimum TU size of 4 × 4 samples was held constant, the maximum TU size was successively increased. The corresponding coding efficiency gains are summarized in Table 3.4. A significant gain of 6.8 % for the entertainment scenario and 8.5 % for the interactive scenario is already obtained by additionally enabling 8 × 8 TUs, a configuration which is conceptionally similar to the High profile of H.264 | MPEG-4 AVC. The bit-rate saving relative to the configuration with 4 × 4 TUs only is further increased to 13.9 % and 17.5 % for the entertainment and interactive application scenarios, respectively, if all transform sizes supported in HEVC are enabled.

Besides the maximum and minimum TU sizes, also the maximum depth of the residual quadtree (RQT) can be chosen by an encoder in HEVC. For investigating its impact on coding efficiency, we enabled all transform sizes, but restricted the maximum RQT depth. In the reference configuration, we set the maximum RQT to the minimum supported value of 0. It should be noted that in this setting, as already discussed above, the partitioning of a CU into multiple PUs implies also the subdivision into four TUs. If the maximum RQT depth is set equal to 1, the subdivision of a CU into four TUs can be selected by the encoder, unless the CU

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00081

Table 3.5 Coding efficiency improvement for successively increasing the maximum depth of the residual quadtree (RQT). The shown average bit-rate savings are measured relative to a configuration in which the RQT depth is equal to 0. All supported CU, PU, and TU sizes are enabled

Residual quadtree depth	Entertainment applications	Interactive applications
RQT depth equal to 1	0.7 %	0.7 %
Maximum supported RQT depth	1.2 %	1.0 %

Table 3.6 Coding efficiency improvement for successively increasing the CTU size and the number of hierarchy levels in the coding tree. The shown average bit-rate savings are measured relative to a configuration with a CTU size of 16 × 16 and a minimum CU size of 8 × 8 luma samples

CTU size and minimum CU size	Entertainment applications	Interactive applications
32 × 32 CTU, 16 × 16 minimum CU	9.2 %	17.4 %
32 × 32 CTU, 8 × 8 minimum CU	12.1 %	20.2 %
64 × 64 CTU, 16 × 16 minimum CU	12.7 %	23.8 %
64 × 64 CTU, 8 × 8 minimum CU	14.9 %	25.5 %

is subdivided for the purpose of signaling intra prediction modes. The results are summarized in Table 3.5. It can be seen that providing the possibility of applying the transform across PU boundaries yields a bit-rate saving of 0.7 % for both application scenarios. By allowing more than one level of the residual quadtree, the coding efficiency can be further increased, even though the improvement is small compared to the impact of other coding options.

In the next experiment, we compared different configurations for the CTU size and the minimum CU size. As reference, we used a configuration with a CTU size of 16 × 16 luma samples and a minimum CU size of 8 × 8 luma samples. This configuration provides similar partitioning modes as the High profile of H.264 | MPEG-4 AVC. The bit-rate savings obtained by successively increasing both the CTU size and the depth of the coding tree are summarized in Table 3.6. A significant gain is already achieved if all block sizes are basically increased by a factor of 2 in horizontal and vertical direction, corresponding to a CTU size of 32 × 32 luma samples and a minimum CU size of 16 × 16 luma samples. By further increasing the depth of the coding tree and the CTU size, the compression performance is further improved for the tested HD sequences.

If we look at the development of video coding standards from H.262 | MPEG-2 Video to HEVC, one key aspect for improving the compression performance was to increase the set of supported block sizes for motion-compensated prediction and transform coding. In the last experiment, we evaluated the associated coding efficiency improvement using the HEVC reference software. It should be noted that we do not compare the different video coding standards, but only investigate the impact of increasing the supported set of block sizes for motion-compensated prediction and transform coding. For all other aspects, the coding tools of HEVC are used in the experiment. As a reference, we used a configuration that is

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00082

Table 3.7 Coding efficiency improvement for increasing the set of PU and TU sizes as has been done in the development of video coding standards. The shown average bit-rates are measured relative to a configuration corresponding to H.262 | MPEG-2 Video (16 × 16 PUs and 8 × 8 TUs only)

Supported PU and TU sizes	Entertainment applications	Interactive applications
H.263 & MPEG-4 Visual	2.8 %	4.3 %
H.264 | MPEG-4 AVC	7.3 %	5.7 %
HEVC	26.6 %	37.6 %

restricted to 16 × 16 PUs and 8 × 8 TUs and thus corresponds to the block sizes supported in H.262 | MPEG-2 AVC. A configuration corresponding to H.263 and MPEG-4 Visual is obtained by additionally allowing PU sizes of 8 × 8 luma samples. For simulating the partitioning features of H.264 | MPEG-4 AVC, we set the maximum CTU size to 16 × 16 luma samples, allowed all subdivisions supported in HEVC, except the asymmetric partitioning modes, and additionally enabled 4 × 4 PUs by slightly modifying the syntax. Even though this configuration differs in some aspects from the actual syntax features of H.264 | MPEG-4 AVC, it still provides a suitable comparison point. Finally, in the HEVC configuration, the maximum set of supported block sizes was enabled. The average bit-rate savings relative to the H.262 | MPEG-2 Video configuration are summarized in Table 3.7. Additionally, diagrams with rate-distortion curves for a representative sequence for both application scenarios are shown in Fig. 3.8. For the tested HD sequences, the increase of partitioning modes from H.262 | MPEG-2 Video to H.264 | MPEG-4 AVC yields bit-rate savings of 7.3 and 5.7 % for the entertainment and interactive scenario, respectively. The experiment indicates that most of the coding efficiency improvements [31] that are obtained by H.264 | MPEG-4 AVC for HD material are related to other coding tools, not the additional partitioning modes. In contrast to that, a significant part of the coding gain of HEVC relative to H.264 | MPEG-4 AVC can be attributed to supporting increased block sizes for motion-compensated prediction and transform coding.

3.3 Picture Partitioning for Packetization and Parallel Processing

As already noted above, both HEVC and H.264 | MPEG-4 AVC follow the same block-based hybrid video coding paradigm. Conceptually, both video coding standards also share the two-layered high-level system design consisting of a video coding layer (VCL) and a network abstraction layer (NAL). The VCL includes all low-level signal processing, including block partitioning, inter- and intra-picture prediction, transform coding, entropy coding, and in-loop filtering. The NAL encapsulates coded data and associated information into NAL units, a logical data packet format that facilitates video transmission over various transport layers.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00083

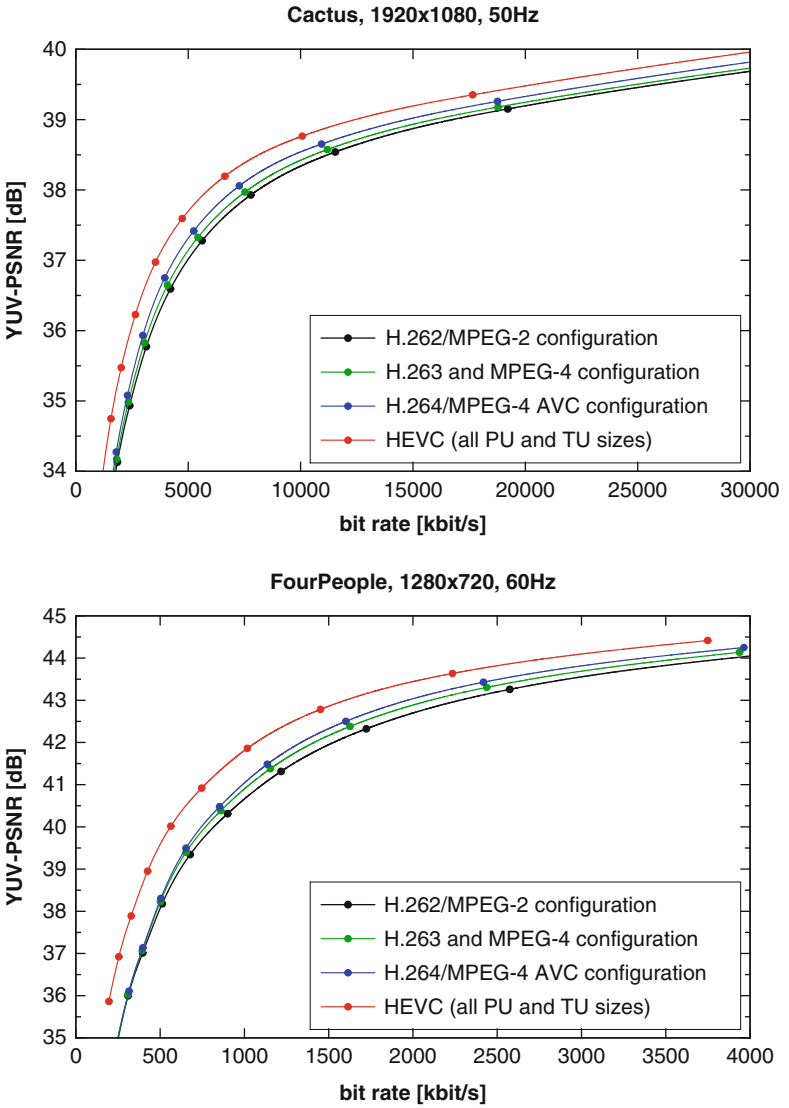


Fig. 3.8 Impact of restricting the PU and TU sizes of HEVC in a way that corresponds to the syntax features of different prior video coding standards. The diagrams shows the corresponding distortion-rate curves for a representative example for the scenario of entertainment applications (*top*) and the scenario of interactive applications (*bottom*)

As in H.264 | MPEG-4 AVC, an HEVC bitstream consists of a number of access units, each including coded data associated with a picture that has a distinct capturing or presentation time. Each access unit is divided into NAL units, including one or more VCL NAL units and zero or more non-VCL NAL units. The VCL

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00084

NAL units consist of coded slices which, as will be discussed in more detail in Sect. 3.3.1 below, represent grouped blocks of samples of the video picture. The non-VCL NAL units contain associated data such as, e.g., parameter set NAL units or supplemental enhancement information (SEI) NAL units. Parameter sets contain data that are essential for the decoding process, while the SEI syntax enables the optional support of supplemental data. Besides the sequence parameter set (SPS) and the picture parameter set (PPS), as known from H.264 | MPEG-4 AVC, HEVC also includes the novel video parameter set (VPS) for conveying additional metadata about the characteristics of the coded video sequences to be used at the systems layer.

Each NAL unit consists of a NAL unit header and a NAL unit payload. The two-byte NAL unit header in HEVC contains information about the type of payload and, similar to the scalable video coding (SVC) extension of H.264 | MPEG-4 AVC, a temporal identifier, which indicates the level in the temporal hierarchical prediction structure. This information can be conveniently accessed by media gateways, also known as media-aware network elements (MANEs), for intelligent, media-aware operations on the stream, such as bitstream thinning using temporal scalability. Note that this can be achieved without the need of parsing the NAL payload data. The payload data is also interleaved with emulation prevention bytes when necessary to ensure that no byte-aligned start code prefix within the NAL payload is emulated. Details about the NAL concept and parameter sets can be found in Chap. 2.

3.3.1 Slices and Their Fragmentation into Slice Segments and Slice Segment Subsets

The high-level segmentation of a picture in HEVC is achieved similar to that in H.264 | MPEG-4 AVC based on the slice concept. The slice concept provides a partitioning of a picture in such a way that each slice is independently decodable from other slices of the same picture, where decoding refers to entropy, residual, and predictive decoding. A slice may consist of a complete picture as well as parts thereof. In HEVC, the minimum block structure unit of a picture contained in a slice is a single coding tree unit (CTU).

Picture partitioning by slices serves the following three purposes:

1. **Error Robustness**: To partition the picture into smaller self-contained entities in order to gain error robustness by the ability to re-synchronize both the decoding and parsing process in case of data losses. This typically implies that the slices are transported packet-wise, i.e., a loss of a transport packet results in a loss of a slice.
2. **MTU Size Matching**: To adapt to the network constraint of maximum transmission unit (MTU) size commonly found in IP networks. Such a packetization scheme is also referred to as MTU size matching and restricts the maximum number of payload bits within a slice regardless of the size of the coded picture.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00085

To keep each slice within this limit and, at the same time, minimize the packetization overhead, the encoder may produce slices with varying number of CTUs within a picture.

3. **Parallel Processing**: To partition the picture into units which can be processed in parallel. This is given by the fact that all slice-based encoding/decoding operations including reconstruction prior to loop filtering can be independently carried out in parallel.

Similar to what is specified in H.264 | MPEG-4 AVC, slices in HEVC consist of an integer number of its minimum building block which, as already noted above, is given by a CTU instead of a macroblock. CTUs in a slice are processed in raster scan order such that each slice of a picture is independently parsable and decodable. This is achieved by terminating the CABAC bitstream at the end of each slice and by breaking CTU dependencies across slice boundaries within a picture such as, e.g., dependencies used for in-picture prediction, context selection, or probability estimation. Due to this reduced exploitation of spatial redundancy, the coding efficiency usually decreases quite substantially with increasing the number of slices used for a picture.

Conceptually, a slice consists of a slice header and the slice data. The slice header provides specific information for the decoding of the slice data, i.e., the coded CTUs within the picture to which the slice belongs. Therefore, the slice header precedes the actual slice data. Note that the overhead of each slice header also contributes to the reduced coding efficiency when using multiple slices, especially at lower bit rates.

During the development of HEVC it turned out that the conventional slice concept, as outlined above and supported by prior video coding standards, is a too rigid concept to properly meet all anticipated needs. In particular, the bit-rate overhead caused by multiple slice headers and the strict breaking of in-picture dependencies at slice boundaries was found to be critical in certain application use cases.

Therefore, HEVC introduces the novel functionality of slice fragmentation at conceptually two distinct levels. For the first level of fragmentation, each slice can be divided into one or more slice segments at the CTU boundaries. The first slice segment of a slice (in CTU raster scan order) is the independent slice segment and includes the full slice (segment) header. The independent slice segment is also often referred to as a regular slice, since it is conceptually equivalent to what is specified in H.264 | MPEG-4 AVC. All subsequent slice segments within a slice (if any) are so-called dependent slice segments with drastically shortened slice segment headers. Note that within the same slice CTU dependencies across slice segment boundaries, both in terms of in-picture prediction and entropy coding, are allowed for dependent slice segments, provided that no further restrictions are given. Slice segments will be discussed in more detail in Sect. 3.3.1.1 below.

The second slice fragmentation level of HEVC is given by the instrument of slice segment subsets, often referred to as substreams. Each slice segment subset contains all coded bits of a subset of CTUs covered by the corresponding slice

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00086

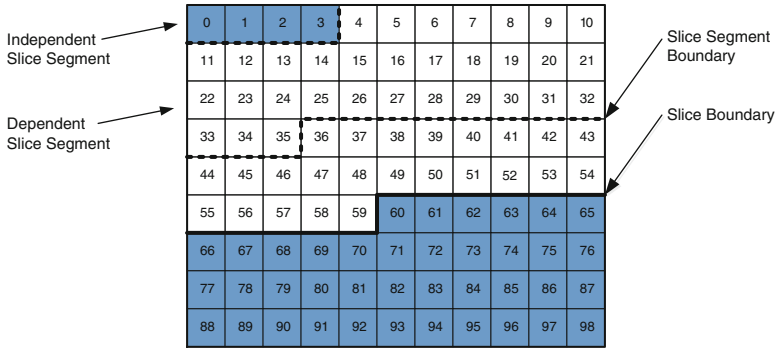


Fig. 3.9 Picture structuring in HEVC using slices, showing the CTUs belonging to independent slice segments (*blue shaded*) and dependent slice segments of two slices of a picture. Note that the slice boundaries are marked by a *solid line*, whereas the boundaries between slice segments of the same slice are marked by *dashed lines*

segment data, and an entry point offset is provided for each (but the first) subset in the corresponding slice segment header for proper subset identification. Subsets are most appropriate for the use together with the novel high-level parallelization tools of HEVC, as will be discussed in Sect. 3.3.2. Slice segment subsets will be presented in Sect. 3.3.1.2, followed by a discussion of the slice segment header in Sect. 3.3.1.3.

3.3.1.1 Slice Segments

The fragmentation of slices by the use of dependent slice segments was first proposed in [34]. According to this fragmentation concept, a slice in HEVC is defined as a set of slice segments, where the first segment of a slice is the independent slice segment, followed by zero or more dependent slice segments, as exemplarily shown in Fig. 3.9.

Dependent slice segments do only contain a minimum set of slice header parameters and its use is indicated by the slice header syntax element `dependent_slice_segment_flag`, as further explained in Sect. 3.3.1.3. As another distinct feature in comparison to regular slices or independent slice segments, dependent slice segments do not break in-picture dependencies across CTU boundaries within the slice to which the segment belongs. Although each dependent slice segment data is conveyed by its own CABAC bitstream and therefore, the CABAC engine needs to be flushed and reset at the segment boundaries, the adapted content of all CABAC context variables are stored at the end of each slice segment (including those of the independent slice segment) in order to be re-used for initialization in the subsequent dependent slice segment. In this way, no coding efficiency penalty is introduced by the use of dependent slice segments except for the additional, rather small amount of slice segment header bits. Note that for each CTU in a slice segment, the syntax

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00087

element `end_of_slice_segment_flag` is signaled to indicate whether the corresponding CTU is the last CTU in the slice segment and, if this is the case, to properly terminate the CABAC bitstream and to perform the above-mentioned storage process for CABAC context variables. For more details about termination and the handling of context memory in HEVC CABAC, please refer to Chap. 8.

Obviously, dependent slice segments do not provide the same error robustness as independent slice segments. In an error-free environment, however, an encoder can choose to fragment a coded picture in potentially many small units and provide them to the network transmission stack before having finished encoding the remainder of the picture, and without incurring the penalty of broken in-picture dependencies.

One use case of dependent slice segments is the reduction of the end-to-end delay for ultra-low delay applications, such as remote video or broadcast contribution. Since the dependent slice segment header only carries a minimum of data, slice segments may also be used as entry points for parallelization techniques. The latter aspect as well as the relation to ultra-low delay requirements is discussed in more detail in Sect. 3.3.4.

In addition, the slice segment concept provides a fragmentation mechanism for bitstream partitioning of over-sized NAL units to comply with the MTU size requirements without incurring substantial coding efficiency losses, as it is usually the case for regular slices. Slice segments are also useful for providing a correct and byte-aligned fragmentation of entropy coded data in such a way that in case of losing a dependent slice segment, the independent slice segment together with all its dependent slice segments preceding the lost segment can be decoded correctly.

3.3.1.2 Slice Segment Subsets

Picture partitioning for the purpose of parallel processing does not necessarily require each resulting partition to be included in an individual NAL unit. Therefore, HEVC provides a fragmentation of coded slice data without the use of additional header data. This is achieved by dividing the slice segment data, preferably that of an independent slice segment, into disjoint subsets of coded CTU data such that the union of all subsets cover the whole slice segment data. Each slice segment subset consists of an individual byte aligned CABAC bitstream, which is terminated by using the special terminating syntax element `end_of_sub_stream_one_bit`. For all but the first substream, the entry point offsets (in bytes) are signaled by the syntax elements `entry_point_offset_minus1[i]` in the corresponding slice segment header, where the length (in bits) of the entry point offset syntax elements is given by the prior signaled `offset_len_minus1` syntax element. Note that the first subset starts with the first byte of the slice segment data, immediately after the slice segment header data, which is considered to be byte 0. It is also important to understand that the entry point signaling is based on the calculation of the byte aligned substream lengths including any potentially necessary emulation prevention bytes. However, without any further externally derived information of the location of the first CTU in each slice segment subset,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00088

no parallel decoding is possible and, therefore, it is quite obvious that there are a number of further restrictions imposed on slice segment subsets, when used together with the new high-level parallelization tools in HEVC, as will be further explained in Sect. 3.3.2.

3.3.1.3 Slice Segment Header

A lot of syntax elements in the HEVC slice segment header are already known from the corresponding slice header syntax in H.264 | MPEG-4 AVC. This subsection gives a brief survey of the main syntax elements in the slice segment header of both independent and dependent slice segments in HEVC by discussing the commonalities and differences in relationship to the slice header syntax of its predecessor.

If the slice segment is not the first slice segment in a picture, which is signaled by the `first_slice_segment_in_pic_flag` at the beginning of the slice segment header, the syntax element `slice_segment_address` determines the address of the first CTU in the slice segment in CTU raster scan order, analogous to the address of the first macroblock in the slice header of H.264 | MPEG-4 AVC. Note that the first CTU of the first (independent) slice segment in a picture is always given as the CTU that covers the luma CTB containing the top left luma sample of the picture.

Each slice segment header refers to the specific picture parameter set that is in use for the picture to which the slice belongs. The corresponding identifier `slice_pic_parameter_set_id` must have the same value for all slices of a given picture. The referred PPS contains further information such as the referred SPS and information controlling the presence of particular syntax elements in the slice segment header.

All above-mentioned syntax elements are present for both independent and dependent slice segment headers. The discrimination between both slice segment types is signaled by the `dependent_slice_segment_flag`, if the corresponding fragmentation functionality is enabled in the PPS. Since for dependent slice segments only a shortened header is transmitted, all values of the remaining slice header syntax elements, excluding the entry point offset signaling for slice segment subsets, are derived from the full slice segment header of the preceding independent slice segment, instead of being explicitly transmitted.

One of these syntax elements unique to the header of independent slice segments is the `slice_type` that indicates the coding type of the slice. This syntax element is also known from H.264 | MPEG-4 AVC and specifies whether the slice is a B slice, a P slice, or an I slice, depending on whether the use of bi-predictive, uni-predictive, or only intra-predictive coding is allowed, respectively.

For all pictures which are not instantaneous decoding refresh (IDR) pictures, the (independent) slice segment header also contains information that allows derivation of the picture order count (POC) of the enclosing picture. The POC allows to identify pictures that needs to be present in the decoded picture buffer (DPB) for

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00089

decoding of remaining pictures, where the set of retained reference pictures is called the reference picture set (RPS). For a proper management of the RPS, HEVC supports a couple of new syntax elements in the (independent) slice segment header that lead to a far more robust approach in case of picture and/or slice losses when compared to the corresponding capabilities of H.264 | MPEG-4 AVC.

Other syntax elements in the (independent) slice segment header are directly related to the decoding and parsing process, such as a flag indicating the use of the temporal motion vector predictor, the slice QP delta, parameters for weighted prediction as well as a flag indicating the use of a specific set of CABAC initialization values for P and B slices.

For in-loop filtering, there are optional flags in the (independent) slice segment header for enabling/disabling the deblocking filter for the current slice or for enabling/disabling the whole in-loop filtering, including sample-adaptive offset (SAO) filtering, across the left and upper boundaries of the current slice.

In order to be future proof in terms of extensibility, the HEVC slice header syntax also includes optional syntax elements that allow for extensions of the slice header by later versions and/or profiles of the standard.

3.3.2 High-Level Parallelization Features

Parallel processing can enable the operation of a video codec in real-time on systems which would not be able to support the codec operation in real-time in a non-parallel execution fashion. Today's hardware architectures inherently support multi-threading also on low power platforms. This is somewhat motivated by the fact that manufacturers of general purpose processors and later also manufactures of low-power ARM based processors started producing multi-core processors when it became more difficult/costly to boost CPU processing power by raising clock speed as they had done prior to the multi-core revolution. The performance gain of multi-threading is only achievable, if the target platform also supports the parallel execution of threads in such a way that multiple hardware execution units such as, e.g., multi-core processor(s) can be used.

Multi-threading in the software context refers to a programming model in which the computation is specified in multiple independent units called threads that share some resources, specifically memory. These threads can run in a time division or in a parallel manner depending on the implementation and underlying hardware. Threads usually communicate via shared memory, and have to use synchronization operations to ensure a proper use of shared resources and to satisfy ordering constraints in the program. Commonly synchronization statements such as locks, semaphores, barriers, and condition variables are used to control the progress among the threads.

The performance gain achieved by multi-threading is measured as speed up in terms of execution time of the program code on a single processing unit divided by the execution time of the program code on multiple p parallel processing units.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00090

Thus, typically the maximum speed-up factor using p processing units is p. Since the actual speed-up is influenced by various characteristics such as synchronization between threads, memory access or memory characteristics, an optimized implementation would avoid main memory access, assuming a hierarchical memory structure, as much as possible and instead rely on cache memory access, as the latter typically has much faster access times compared to the main memory albeit at smaller memory sizes.

Different parallelization techniques have been introduced for improved utilization of computational resources in the implementation of video coding standards. In the following only the most important ones are mentioned:

- **Picture-Level Parallelization**: Picture-level parallelism consists of processing multiple pictures at the same time provided that the temporal dependencies for motion-compensated prediction are satisfied. Picture-level parallelism is often sufficient for multi-core systems with a few cores. Because it is relatively simple to implement and does not incur coding efficiency losses, it has become the state-of-the art for software-based implementations of H.264 | MPEG-4 AVC. However, picture-level parallelism has a number of limitations. First, the parallelization scalability is determined by the lengths of the motion vectors and/or the size of the underlying group of pictures (GOP). Second, the workload of each core may be imbalanced because the picture encoding/decoding times can vary significantly. Finally, picture-level parallelism increases the processing frame rate but does not improve latency.
- **Slice-Level Parallelization**: In HEVC and H.264 | MPEG-4 AVC, each picture can be partitioned into slices, as described in Sect. 3.3.1. All (regular) slices within a picture are independent from each other except for potential dependencies regarding cross-slice border in-loop filtering. Therefore, slices can be used for parallel processing. Slice-level parallelism, however, has a number of disadvantages. Although slices are completely independent from each other in terms of prediction, transform and entropy coding, in-loop filtering may be applied across slice boundaries. For H.264 | MPEG-4 AVC it may be required to perform deblocking of the complete picture using a single processing unit, whereas HEVC, in principle, allows in-loop filtering to be performed on CTU rows in parallel. Moreover, as already mentioned above, multiple slices reduce the coding efficiency significantly due to the restrictions of in-picture prediction and entropy coding across slice boundaries. Due to these disadvantages, exploiting slice-level parallelism is only advisable when the number of slices per picture is strictly limited [33].
- **Block-Level Parallelization**: In hardware-based implementations of H.264 | MPEG-4 AVC, for example, a macroblock-level pipeline is very widely used. This kind of block-level parallelization technique is based on using heterogenous processing cores, where one core is dedicated for entropy coding, one for in-loop filtering, one for intra prediction and so on. In this way, macroblocks will be processed concurrently on the different cores. Note, however, that efficient parallel processing of macroblocks may require an elaborate scheduling

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00091

algorithm for determining the order of macroblock processing, given their multiple spatial dependencies. Another approach of block-level parallelism is given by the wavefront scheduling approach [40], where macroblocks are grouped in a wavefront-like manner to ensure that a sufficient number of macroblocks are available, as dictated by the spatial dependencies between adjacent macroblocks. At the same time, all macroblocks belonging to the same "wavefront" can be processed concurrently. Furthermore, macroblocks of different pictures can be processed in parallel provided that the temporal dependencies due to motion-compensated prediction are handled correctly [28]. Entropy decoding, however, can only be parallelized at the slice level and therefore it has to be decoupled from macroblock or CTU reconstruction. Although this approach can scale up to multi-core architectures, it has some limitations too. First, the decoupling of entropy decoding and reconstruction increases the memory usage. Furthermore, this strategy only reduces the decoding time of a picture in the reconstruction stage but not in the entropy decoding stage. Consequently, a single-threaded entropy decoding step itself may be the bottleneck and the limiting factor of the overall throughput.

In order to overcome the limitations of the parallelization strategies employed in H.264 | MPEG-4 AVC, HEVC provides VCL-based coding tools that are specifically designed to enable processing on high-level parallel architectures. Two new tools aiming at facilitating high-level parallel processing have been included in the HEVC standard [9, 10, 18]:

- **Wavefront Parallel Processing (WPP)**: A parallel processing approach along the wavefront scheduling principle, which is based on a partitioning of the picture into CTU rows such that the dependencies between CTUs of different partitions, both in terms of predictive coding and entropy coding, are preserved to a large extent.
- **Tiles**: A picture partitioning mechanism similar to slices, which is based on a flexible subdivision of the picture into rectangular regions of CTUs such that coding dependencies between CTUs of different partitions are prohibited.

Both of these tools allow subdivision of each picture into multiple partitions that can be processed in parallel. Each partition contains an integer number of CTUs that may or may not have dependencies on CTUs of other partitions. When WPP or tiles are enabled, typically for each partition a separate slice segment subset is used such that the corresponding entry point offsets (in the slice segment header) indicate the start positions of all picture partition substreams (except for the first substream) in the slice segment. This is necessary for each core to immediately access the partition it has been assigned to decode. More details about parallel partition access are given in Sect. 3.3.2.3 below.

In the HEVC Main, Main10 and Main Still Picture profile, only one of the tools can be used at the same time, although the entry point signaling design would allow for a co-existence in future profiles.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00092

Although the following subsections discuss the high-level parallelization tools mainly from a decoder point-of-view, both WPP and tiles may be used for parallelization of encoders as well.

3.3.2.1 Tiles

When tiles are enabled in HEVC, the picture is divided into rectangular-shaped groups of CTUs separated by vertical and/or horizontal boundaries [9, 18, 29]. The PPS syntax element `tiles_enabled_flag` indicates the use of tiles. The tile approach has some similarities to slice groups in H.264 | MPEG-4 AVC using the feature of flexible macroblock ordering (FMO) with slice group map type 2.

The number of tiles and the location of their boundaries can be defined for the entire sequence or changed from picture to picture. This is achieved in the PPS by signaling parameters such as `num_tile_columns_minus1`, `num_tile_rows_ minus1`, `uniform_spacing_flag`, `column_width _minus1` and `row_ height_minus1`. With these syntax elements the number of tiles, as well as the picture partitioning into tiles can be defined. Since the tile signaling is included in the PPS, this structure may change on a per picture basis. This has the benefit that an encoder can use different ways of arranging the tiles in different pictures in order to control the load balancing between cores used for encoding/decoding. For example, if a region of a picture should require more processing resources, such a region may be subdivided into more tiles than other regions which may require less encoding/decoding resources. But the required encoder/decoder resource management, which determines the tile structure, needs to be applied before the actual encoding process.

Tile boundaries, similarly to slice boundaries, do break parsing and prediction dependencies so that a tile can be processed independently, but the in-loop (deblocking and SAO) filters can still cross tile boundaries in order to optionally prevent tile border artifacts. This functionality is controlled by the `loop_filter_across_ tiles_enabled_flag` syntax element in the PPS.

The fact that a tile can be processed independently from other tiles in a picture makes tiles also very suitable for lossy transmission environments, if tiles of a picture are transported in different packets.

Tiles do change the regular scan order of CTUs from picture-based raster scan order (i.e., CTUs are processed row-wise from left to right) to a tile-based raster scan order, of which an example is shown in Fig. 3.10. Since the tile scan order itself may be different from the picture raster scan order used in typical picture processing systems, decoders may process the CTUs of tiles still in a picture-based raster scan fashion not exploiting any parallelization. Note that this can be achieved by storing and reloading the values of all CABAC context variables for each tile at the tile boundaries while moving from one CTU row to the next.

Constraints are set on the relationship between slices, slice segments and tiles. At least one of the following conditions shall be true for each slice and tile in a picture: All CTUs in a slice belong to the same tile, or all CTUs in a tile belong to the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00093

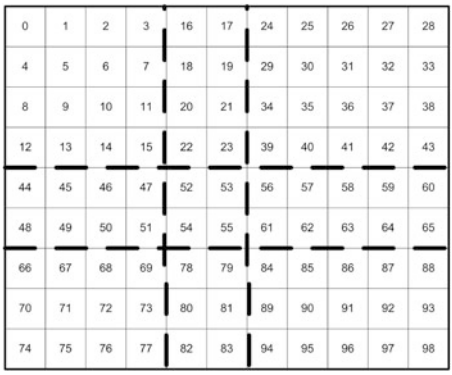


Fig. 3.10 Tile-based raster scan order of CTUs with nine tiles of different sizes in the picture. Note that the tile boundaries are marked with *bold dashed lines*

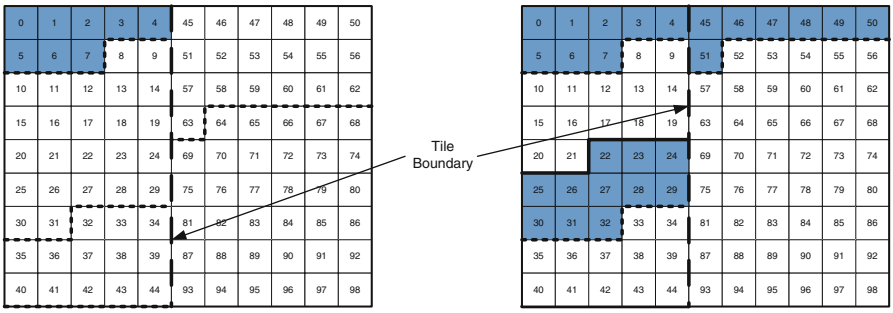


Fig. 3.11 Two different ways of fragmenting an exemplary tile partitioning using two tiles into slices and slice segments. *Left*: One slice including four dependent slice segments. *Right*: Three slices, each containing one dependent slice segment. Note that the tile boundaries are marked with *bold dashed lines*, the slice boundaries with *bold solid lines*, and the boundaries between slice segments with *dotted lines*. The CTUs belonging to the independent slice segments are *blue shaded*

same slice. For each slice segment and tile at least one of the following conditions shall be true in a picture: All CTUs in a slice segment belong to the same tile, or all CTUs in a tile belong to the same slice segment. Note that as a consequence of these constraints, a slice or slice segment whose starting point does not coincide with the starting point of a tile cannot span multiple tiles. An example of this situation is given in Fig. 3.11, which shows the same partitioning of a picture into two tiles with two different ways of fragmentation into slices and slice segments.

Tiles do not require communication between processors for CTU-level entropy decoding and reconstruction, but communication is needed if in-loop filtering stages operate in the cross tile-border filtering mode. Although the cross tile-border filtering mode may be switched off to avoid data exchange between processors, it can result in visual artifacts at tile boundaries.

Compared to slices, tiles usually provide a better coding efficiency since reduced spatial distances in tiles lead to a potentially higher exploitation of spatial correlations between samples within a tile. Furthermore, the use of tiles may reduce slice header overhead, at least in cases where not exactly one slice per tile

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00094

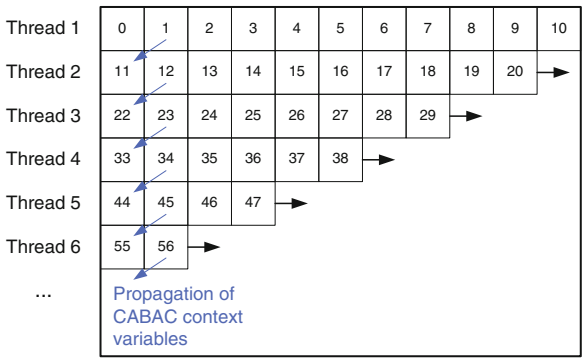


Fig. 3.12 Wavefront parallel processing along CTU rows and propagation of CABAC context variables from CTU1 to CTU11, from CTU12 to CTU22, . . . , from CTU45 to CTU55, and so on

is used. But, similar to slices, the coding efficiency loss typically increases with the number of tiles, due to the breaking of dependencies along tile boundaries and the re-initialization of all CABAC context variables at the beginning of each tile.

3.3.2.2 Wavefront Parallel Processing (WPP)

When wavefront parallel processing (WPP) is enabled in HEVC, each CTU row of a picture constitutes a separate partition [10, 18]. WPP is enabled/disabled by the PPS syntax element `entropy_coding_ sync_enabled_flag`. As shown in Fig. 3.12, in WPP mode each CTU row is processed relative to its preceding CTU row by using a delay of two consecutive CTUs. In this way, no dependencies between consecutive CTU rows are broken at the partition boundaries except for the CABAC context variables at the end of each CTU row. To mitigate the potential loss in coding efficiency that would result from the conventional CABAC initialization at the starting point of each CTU row, the content of the (partially) adapted CABAC context variables are propagated from the encoded/decoded second CTU of the preceding CTU row to the first CTU of the current CTU row, as shown in Fig. 3.12. As a result, the coding efficiency losses introduced by WPP are relatively small compared to the case of a picture encoding using no WPP but with otherwise identical settings [6, 10]. Also, WPP does not change the regular raster scan order of CTUs. Furthermore, by using a relatively simple CABAC transcoding technique, a WPP bitstream can be transcoded to or from a non-WPP bitstream without any change to the picture reconstruction process [7].

When WPP is enabled, a number of threads up to the number of CTU rows in a picture can work in parallel to process the individual CTU rows, where the number of CTU rows depends on the ratio of the picture height in luma samples and the luma CTB size in either width or height.

By using wavefront parallel processing in a decoder, each decoding thread processes a single CTU row of the picture. Since the CTU dependencies need to be maintained between the decoding threads, the scheduling of the thread processing must be organized in such a way that for each CTU the decoding of its top right

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00095

neighboring CTU in the preceding CTU row must have been finished. In other words, at any time the thread processing of the preceding CTU row must have processed two consecutive CTUs more than the thread processing of the current CTU row, which results in a "wavefront" of CTUs rolling from the top left to the bottom right corner of the picture, as illustrated in Fig. 3.12. This is why the wavefront dependencies do not allow all threads of processing CTU rows to start decoding simultaneously. Consequently, the row-wise CTU processing threads cannot finish decoding at the same time at the end of each row. This introduces parallelization inefficiencies, referred to as ramping inefficiencies, that become more evident with an increasing number of threads being used. Additional pipelining issues might arise because of stalls from an inefficient load balancing of CTUs. For example, a slow CTU in one CTU row could cause stalls in the processing of subsequent CTU rows.

As an additional small overhead in processing, WPP requires to store the content of all CABAC context variables after having finished encoding/decoding of the second CTU in each CTU row. Beyond that, however, WPP does not require any extra handling of partition borders to preserve the dependencies utilized by entropy encoding/decoding, in-picture prediction or in-loop filtering. An example WPP scheme for executing all HEVC decoder operations within the hybrid video coding loop can be found in [6].

The header overhead associated with WPP can be kept small and may consist only of signaling the partition entry point offsets via slice segment subsets or, alternatively, the reduced slice segment header of dependent slice segments, as discussed in Sect. 3.3.2.3 below. Together with the minor coding efficiency loss due to the above-mentioned CABAC re-initialization by propagation of context variables at the partition starting points, the resulting overall overhead of a WPP bitstream is small compared to a non-parallel bitstream, while enabling a fair amount of parallelism that scales with the picture resolution.

In order to maintain bitstream conformance for the CTU row partitioning approach, a constraint has been set on the presence of slices and slice segments in a CTU row. According to this constraint, it is required that the last CTU of a slice or a slice segment that does not start with the first CTU of a CTU row belongs to the same CTU row as the first CTU in the same slice or slice segment.

As already mentioned above, the scalability of wavefront parallel processing is limited by the reduced number of independent CTUs at the beginning and at the end of each picture. To overcome this limitation and increase the parallelization scalability of WPP, a technique called overlapped wavefront (OWF) has been proposed in [2, 6], which is not part of the HEVC standard, but may be implemented by using additional constraints on the encoding process. By using OWF, multiple pictures can be decoded simultaneously such that a more constant parallelization gain can be obtained. An in-depth analysis of the parallelization tools included in HEVC has shown that when WPP is combined with the OWF algorithm, it provides a better parallelization scalability than tiles. For quantitative details of the comparison of WPP and tiles as well as the overlapping wavefront approach, the reader is referred to the discussions and results in [6].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00096

3.3.2.3 Bitstream Access for Parallel Decoding

In order to provide access to the individual partitions of the high-level parallelization tools as presented above, both slice segments and slice segment subsets can be used. For the latter, the very efficient entry point signaling in the slice segment header is available. However, as already noted above, for each subset except for the first in a slice segment, the location of the first CTU is not known and, therefore, needs to be derived by external means. This is why the following additional constraints are given on the use of slice segment subsets for tiles and WPP, respectively.

For tiles, each subset of a slice segment shall be fully contained in one tile and the number of subsets shall be equal to the number of tiles that do have at least one CTU in the given slice segment. This means that a subset cannot cover more than one tile and in case, the corresponding slice segment covers more than a single tile, each subset starts with the first CTU in one tile and ends with the last CTU in the same tile. In any case, the address of the first CTU of each subset can be derived from the tile pattern signaled in the PPS. Note that, as already mentioned above, in the case where a slice segment covers more than one tile, all tiles must be fully contained in the same slice segment.

Equivalently for WPP, a subset cannot cover more than one CTU row and in case, the enclosing slice segment covers more than one CTU row, each subset in this slice segment starts with the first CTU in a CTU row and ends with the last CTU in the same CTU row with the possible exception of the last subset. Thus, for every subset the address of its first CTU is equal to the first CTU of a CTU row and each subset, possibly except the last subset, spans a whole CTU row in the given slice segment.

In the two cases where there is only a single tile or a single CTU row or less contained in a slice segment, the offset signaling is not present in the corresponding slice segment header.

As an alternative to slice segment subsets, the use of slice segments, especially dependent slice segments due to its comparably small header overhead may be equally appropriate for the partition access. Since the dependent slice segment header also contains explicitly the address of its first CTU, no further restrictions on the use of dependent slice segments together with tiles and WPP are given beyond the ones already mentioned in the previous subsections.

3.3.3 Support for Ultra-Low Delay Applications

HEVC introduces a new hypothetical reference decoder (HRD) processing concept. Details and fundamentals about the HRD process are given in [8] and Chap. 2. The new concept is defined alternatively to the one based on access units, which is the concept of sub-pictures called decoding units (DU). Those sub-pictures may get assigned an additional decoding timestamp by the DU information SEI message. This timing can be used to indicate additionally to the picture timing SEI, which indicates the timing for coded picture buffer (CPB) removal and decoded picture

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00097

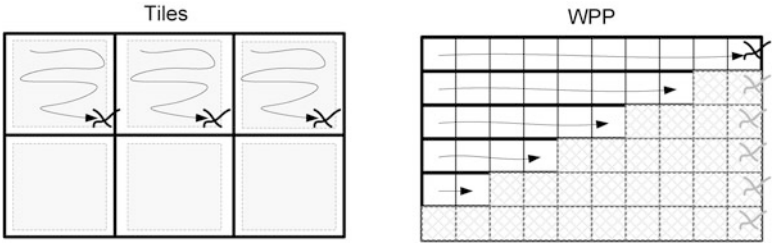


Fig. 3.13 Ultra-low delay operation mode combined with high-level parallelism tools of tiles (*left*) and WPP (*right*) with end of DU indicated by crosses

buffer (DPB) removal for the whole access unit, a timestamp for each DU. The DU timing gives a separate, possibly earlier timing for decoding of the DU compared to the whole picture. This has the merits that parts of the picture may be encoded by the sender, transmitted to the receiver and decoded at the receiver earlier than if the whole access unit is processed at once. This means that this indication of earlier decoding times of sub-pictures allows to reduce the whole end-to-end delay of a capturing, coding, transmission, decoding and presentation chain [8].

In ultra-low delay applications, such as remote-video or broadcast contribution where delays below picture durations are required, the encoder needs to output a picture partition in the form of a DU to the transmission chain as soon as encoding is finished.

HEVC further provides the ability of using high-level parallelization techniques in order to reduce processing demands in multiprocessing unit environments. Therefore, HEVC allows for subdividing the picture into tiles or WPP substreams, i.e., rows of CTUs as described above in Sect. 3.3.2. Either of the methods may be used with ultra-low latency operations, where the WPP case can only be achieved using dependent slice segments, as discussed in Sect. 3.3.1.

The ultra-low latency mode for high-level parallelism is shown in Fig. 3.13. The left part of the Fig. 3.13 shows the coding process of tiles, where the first three tiles (marked by cross) are bound to the same decoding unit. In the right part of Fig. 3.13, six CTU rows are provided, where each CTU row belongs to a single decoding unit, each consisting of a single slice segment (and marked by a cross).

3.3.4 Summary of High-Level Parallelization Tools

It is clear from the previous sections that tiles and WPP have different pros and cons.

WPP is generally well-suited for the parallelization of the encoder and decoder because it allows a high number of picture partitions with relatively low coding-efficiency losses. Additionally, WPP does not need an additional pass of in-loop filtering in comparison to slices and tiles [41]. WPP can also be used for low-delay applications, especially those requiring sub-picture delay (also called ultra-low delay), as described in Sect. 3.3.3.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00098

Tiles are also well-suited for a general parallelization of encoder and decoder. The amount of parallelism is not fixed, as is the case for WPP, allowing the encoder to adjust the number of tiles according to its computational resources. Because tiles can be used to subdivide the picture into multiple rectangles spanning the picture horizontally and vertically, they are also better suited for region of interest (ROI) coding than, e.g., slices. In conversational applications, for example, tiles in combination with a tracking algorithm can be used to dynamically adjust the size and error protection of the ROIs. Tiles are also good for applications where coding tasks need to be distributed onto different hardware machines/systems, since each tile can be, assuming cross tile-border filtering is done separately, processed relatively independent from other tiles within the picture and therefore, in this case require a minimum of communication between the threads processing the individual tiles. However, tiles usually come along with the cost of a significant overhead in bit rate.

In order to simplify implementations of the standard, HEVC does not allow the use of tiles and WPP simultaneously in the same compressed video sequence. It may be interesting, however, to allow some combination of these tools in the future. For instance, it could be necessary to divide an ultra-high definition video signal into sub-pictures using tiles with WPP inside each sub-picture, to enable real-time encoding/decoding.

3.4 Conclusions

The nested quadtree-based block partitioning as a distinguished feature of HEVC has been presented. It has been shown that more than half of the coding efficiency improvements of HEVC relative to H.264 | MPEG-4 AVC for HD video material is obtained solely by introducing this flexible block partitioning concept for improved prediction and transform coding. Furthermore, it has been elaborated on how the two novel picture partitioning features of tiles and wavefront parallel processing can improve the parallel-processing friendliness of HEVC in order to meet the increased demand in computational complexity. Finally, it has also been discussed how the new concept of dependent slice segments in HEVC can facilitate ultra-low delay processing on a sub-picture level without substantial loss in coding efficiency, when used together with the aforementioned novel picture partitioning tools.

References

1. Ahmed N, Natarajan T, Rao KR (1974) Discrete cosine transform. IEEE Trans Comput C-23:90–93
2. Alvarez-Mesa M, George V, Chi CC, Schierl T (2012) Improving parallelization efficiency of WPP using overlapped wavefront, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0425, Stockholm, July 2012

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00099

3. Bjøntegaard, G (2001) Calculation of average PSNR differences between RD curves. ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001
4. Chen P, Ye Y, Karczewicz M (2008) Video coding using extended block sizes. ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG- AJ23, San Diego, Oct. 2008
5. Chou PA, Lookabaugh T, Gray RM (1989) Optimal pruning with applications to tree-structured source coding and modeling. IEEE Trans Inf Theory 35:299–315
6. Chi CC, Alvarez-Mesa M, Juurlink B, Clare G, Henry F, Pateux S, Schierl T (2012) Parallel scalability and efficiency of HEVC parallelization approaches. IEEE Trans Circuits Syst Video Technol 22:1827–1838
7. Clare G, Henry F (2012) An HEVC transcoder converting non-parallel bitstreams to/from WPP, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC- J0032, Stockholm, July 2012
8. Deshpande S, Hannuksela MM, Kazui K, Schierl T (2013) An improved hypothetical reference decoder for HEVC. In Proc. SPIE. 8666, Visual Information Processing and Communication IV, no. 866608, Feb. 2013
9. Fuldseth A, Horowitz M, Xu S, Zhou M (2011) Tiles, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E408, Geneva, Mar. 2011
10. Henry F, Pateux S (2011) Wavefront parallel processing, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196, Geneva, Mar. 2011
11. ISO/IEC 11172-2:1993 Coding of moving pictures and associated audio information for digital storage media at up to about 1.5 Mbit/s – Part 2: Video. (MPEG-1)
12. ISO/IEC 14496-2:1999 Coding of audio-visual objects – Part 2: Video. (MPEG-4 Visual)
13. ITU-R Rec. BT.2020 (2012) Parameter values for ultra-high definition television systems for production and international programme exchange.
14. ITU-T Rec. H.261 (1993) Video codec for audiovisual services at $p \times 64$ kbit/s. 3rd edn.
15. ITU-T Rec. H.263 (2005) Video coding for low bit rate communication. 3rd edn.
16. ITU-T Rec. H.262 and ISO/IEC 13818-2 (2000) Generic coding of moving pictures and associated audio information: Video. (MPEG-2 Video), 2nd edn
17. ITU-T Rec. H.264 and ISO/IEC 14496-10 (2012) Advanced video coding. 7th edn.
18. ITU-T Rec. H.265 and ISO/IEC 23008-10 (2013) High efficiency video coding
19. F. Bossen (2012) Common conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H1100, San Jose, Feb. 2012
20. Joint Collaborative Team on Video Coding (2012) HM 10.1 Reference Software. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-10.1/
21. Kim I-K, Min JM, Lee T, Han W-J, Park JH (2012) Block partitioning structure in the HEVC standard. IEEE Trans Circuits Syst Video Technol 22:1697–1706
22. Kondo K, Suzuki T (2012) AHG7: Level definition to limit memory bandwidth of MC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0106, Geneva, Apr.-May 2012
23. Lee T, Chen J, Han, W-J (2010) TE12.1: Experimental results of transform unit quadtree/2-level test, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C200, Guangzhou, Oct. 2010
24. Li B, Xu J,Wu F, Sullivan G J, Li H (2010) Redundancy reduction in CBF and merging coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C277, Guangzhou, Oct. 2010
25. Marpe D, Schwarz H, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H, Nguyen T, Oudin S, Siekmann M, Sühring K, Winken M, Wiegand T (2010) Video compression using quadtrees, leaf merging and novel techniques for motion representation and entropy coding. IEEE Trans Circuits Syst Video Technol 20:1676–1687
26. Ma S, Kuo C-CJ (2007) High-definition video coding with super-macroblocks. In: Proceedings of visual communications and image processing, vol. 6508
27. McCann K, Han W-J, Kim I-K, Min JH, Alshina E, Alshin A, Lee T, Chen J, Seregin V, Lee S, Hong YM, Cheon MS, Shlyakhov N (2010) Samsung's response to the Call for Proposals on Video Compression Technology, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A124, Dresden, Apr. 2010

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00100

28. Meenderinck C, Azevedo A, Alvarez M, Juurlink B, Ramirez A (2009) Parallel scalability of video decoders. J Signal Process Syst 57:173–194
29. Misra K, Segall A, Horowitz M, Xu S, Fuldseth A, Zhou M (2013) An overview of tiles in HEVC. IEEE J Sel Topics Signal Process 7:969–977
30. Minezawa A, Li B, Sugimoto K, Sekiguchi S, Xu J (2011) Proposed fix on CBF flag signaling, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G444, Geneva, Nov. 2011
31. Ohm J-R, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards – including High Efficiency Video Coding (HEVC). IEEE Trans Circuits Syst Video Technol 22:1669–1684
32. Panusopone K, Fang X, Wang L (2011) Proposal on RQT root location, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E364, Geneva, Mar. 2011
33. RoitzschM(2007) Slice-balancing H.264 video encoding for improved scalability of multicore decoding. In: Proceedings of the 7th ACM IEEE International conference on Embedded Software, pp 269–278
34. Schierl T, George V, Henkel A, Marpe D (2012) Dependent slices, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0229, Geneva, Apr.-May 2012
35. Schwarz H, Wiegand T (2001) Tree-structured macroblock partition, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-O17, Pattaya, Dec. 2001
36. Schwarz H, Marpe D, Wiegand T (2007) Overview of the scalable video coding extension of the H.264/AVC standard. IEEE Trans Circuits Syst Video Technol 17:1103–1120
37. Sekiguchi S, Yamagishi S (2009) On coding efficiency with extended block sizes for UHDTV. MPEG document M16019
38. Siekmann M, Schwarz H, Bross B, Marpe D,Wiegand T (2011) Fast encoder control for RQT, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E425, Geneva, Mar. 2011
39. Sullivan GJ, Wiegand T (1998) Rate-distortion optimization for video compression. IEEE Signal Process Mag 15:74–90
40. Van der Tol EB, Jaspers EGT, Gelderblom RH (2003) Mapping of H.264 decoding on a multiprocessor architecture. In: *Proc. SPIE.* 5022, Image and Video Communications and Processing 2003, 707–718, May 2003
41. Viéron J, Thiesse J-M (2012) On tiles and wavefront tools for parallelism, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0198, Geneva, Apr.-May 2012
42. Wiegand T, Schwarz H (2011) Source coding: Part I of fundamentals of source and video coding. Found Trends Signal Process 4:1–222
43. Wiegand T, Schwarz H, Joch A, Kossentini F, Sullivan G (2003) Rate-constrained coder control and comparison of video coding standards. IEEE Trans Circuits Syst Video Technol 13:688–703
44. Winken M, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H,Marpe D, Oudin S, Preiss M, Schwarz H, Siekmann M, Suehring K, Wiegand T (2010) Video coding technology proposal by Fraunhofer HHI, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A116, Dresden, Apr. 2010

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00101

Chapter 4
Intra-Picture Prediction in HEVC

Jani Lainema and Woo-Jin Han

Abstract The intra prediction framework of HEVC consists of three steps: reference sample array construction, sample prediction, and post-processing. All the three steps have been designed to achieve high coding efficiency while minimizing the computational requirements in both the encoder and decoder. The set of defined prediction modes consists of methods modeling various types of content typically present in video and still images. The HEVC angular prediction provides high-fidelity predictors for objects with directional structures, and the additional planar and DC prediction modes can effectively model smooth image areas.

4.1 Introduction

The HEVC intra prediction methods can be classified in two categories. Angular prediction methods [17] form the first category and provide the codec with a possibility to accurately model structures with directional edges. The methods in the second category, namely planar prediction [7] and DC prediction, provide predictors estimating smooth image content. The total number of intra prediction modes supported by HEVC is 35 as listed in Table 4.1.

All the HEVC intra prediction modes use reference samples from the adjacent reconstructed blocks as illustrated in Fig. 4.1. As the reconstruction is performed at transform block granularity, also the intra prediction is operated at the selected transform block size ranging from 4×4 to 32×32 samples. HEVC allows usage

J. Lainema (✉)
Nokia Research Center, Tampere, Finland
e-mail: jani.lainema@nokia.com

W.-J. Han
Gachon University, Seongnam-si, Gyeonggi-do, Korea
e-mail: hurumi@gmail.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_4
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00102

Table 4.1 Relationship between intra prediction mode number and associated names

Intra prediction mode number	Associated names
0	INTRA_PLANAR
1	INTRA_DC
2 . . . 34	INTRA_ANGULAR[i], i = 2 . . . 34

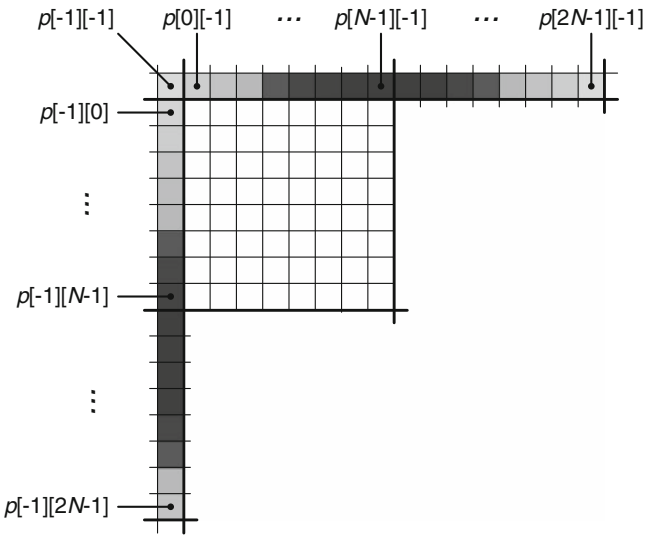


Fig. 4.1 An example of reference samples $p[x][-1]$, $p[-1][y]$ HEVC intra prediction uses for a block of size $N \times N$ samples

of all the defined prediction modes for all the block sizes. Due to the very large amount of different block size—prediction mode combinations all the prediction modes have been designed in a way to allow easy algorithmic implementations for arbitrary block sizes and prediction directions as discussed in the following sections.

In order to improve the likelihood of finding good prediction candidates, HEVC supports different filtering alternatives for pre-processing the reference samples prior to applying those in the actual intra prediction process. Similarly, some of the prediction modes include a post-processing step to refine the sample surface continuity on the block boundaries as described in the following sections. Typical characteristics of the different intra prediction modes are illustrated in Fig. 4.2. The figure represents the final predictions obtained with all the 35 HEVC intra prediction modes when using reference samples shown in Fig. 4.1 as input. Effects of the post-processing are visible for the DC prediction and angular modes 10 and 26 for which the prediction block boundaries include a component from the neighboring sample values.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00103

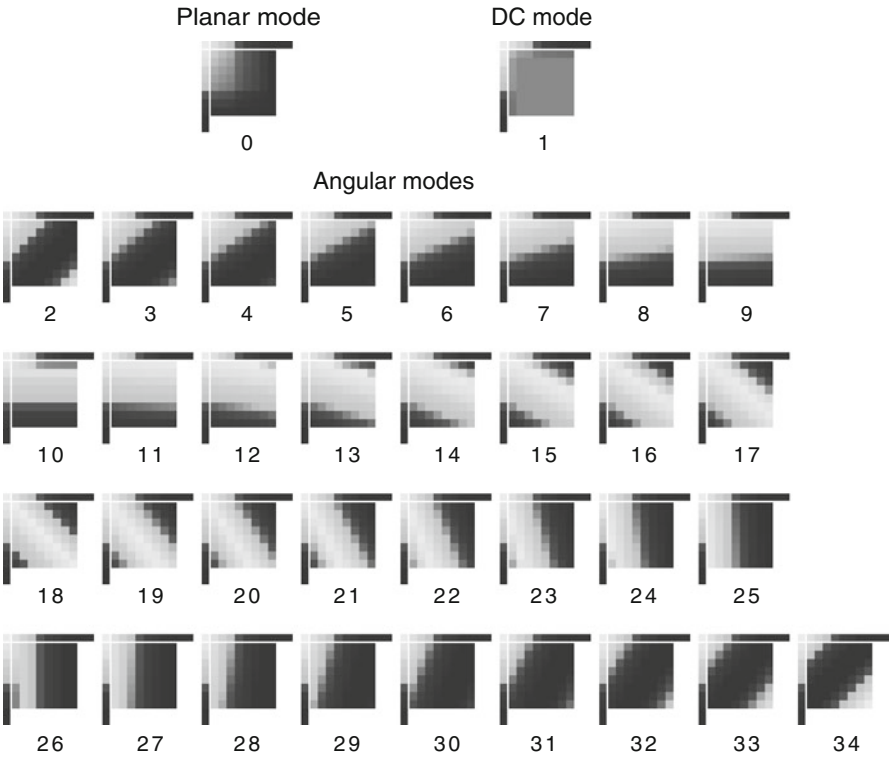


Fig. 4.2 Examples of 8×8 luma prediction blocks generated with all the HEVC intra prediction modes. Effects of the prediction post-processing can be seen on the top and left borders of the DC prediction (mode 1), top border of horizontal mode 10 and left border of vertical mode 26

4.2 Reference Sample Generation

The intra sample prediction in HEVC is performed by extrapolating sample values from the reconstructed reference samples as defined by the selected intra prediction mode. Compared to the H.264/AVC, HEVC introduces a reference sample substitution process which allows HEVC to use the complete set of intra prediction modes regardless of the availability of the neighboring reference samples. In addition, there is an adaptive filtering process that can pre-filter the reference samples according to the intra prediction mode, block size and directionality to increase the diversity of the available predictors.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00104

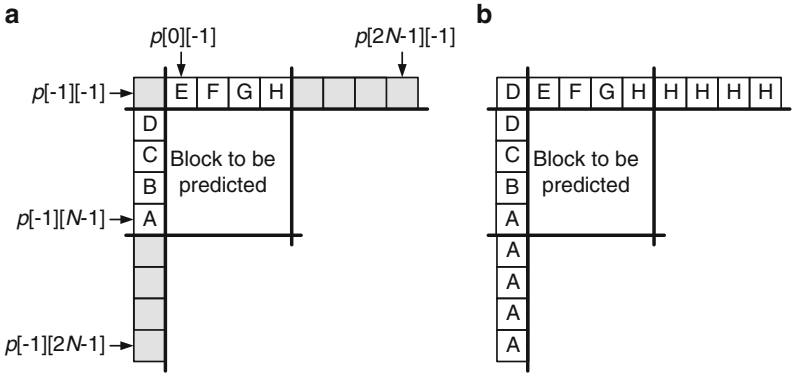


Fig. 4.3 An example of reference sample substitution process. Non-available reference samples are marked as *grey*: (**a**) reference samples before the substitution process (**b**) reference samples after the substitution process

4.2.1 Reference Sample Substitution

Some or all of the reference samples may not be available for prediction due to several reasons. For example, samples outside of the picture, slice or tile are considered unavailable for prediction. In addition, when constrained intra prediction is enabled, reference samples belonging to inter-predicted PUs are omitted in order to avoid error propagation from potentially erroneously received and reconstructed prior pictures. As opposed to H.264/AVC which allows only DC prediction to be used in these cases, HEVC allows the use of all its prediction modes after substituting the non-available reference samples.

For the extreme case with none of the reference samples available, all the reference samples are substituted by a nominal average sample value for a given bit depth (e.g., 128 for 8-bit data). If there is at least one reference sample marked as available for intra prediction, the unavailable reference samples are substituted by using the available reference samples. The unavailable reference samples are substituted by scanning the reference samples in clock-wise direction and using the latest available sample value for the unavailable ones. More specifically, the process is defined as follows:

1. When $p[-1][2N-1]$ is not available, it is substituted by the first encountered available reference sample when scanning the samples in the order of $p[-1][2N-2]$, $\ldots$, $p[-1][-1]$, followed by $p[0][-1]$, $\ldots$, $p[2N-1][-1]$.
2. All non-available reference samples of $p[-1][y]$ with $y = 2N-2 \ldots -1$ are substituted by the reference sample below $p[-1][y+1]$.
3. All non-available reference samples of p$[x][-1]$ with x $= 0 \ldots 2N-1$ are substituted by the reference sample left $p[x-1][-1]$.

Figure 4.3 shows an example of reference sample substitution.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00105

4.2.2 Filtering Process of Reference Samples

The reference samples used by HEVC intra prediction are conditionally filtered by a smoothing filter similarly to what was done in 8×8 intra prediction of H.264/AVC. The intention of this processing is to improve visual appearance of the prediction block by avoiding steps in the values of reference samples that could potentially generate unwanted directional edges to the prediction block. For the optimal usage of the smoothing filter, the decision to apply the filter is done based on the selected intra prediction mode and size of the prediction block.

If DC prediction is used or if the prediction block is of size 4×4 the smoothing filter is switched off. For other cases, block size and directionality of the prediction are used to decide whether the smoothing filter is applied. For 8×8 prediction blocks, the smoothing filter is only applied for the exactly diagonal directions (angular modes 2, 18 and 34). For 16×16 prediction blocks, the smoothing filter is applied to most cases except the near-horizontal and near-vertical directions (angular modes 9, 10, 11, 25, 26 and 27). For 32×32 prediction blocks, only exactly vertical and exactly horizontal directions (angular modes 10 and 26) are excluded from the smoothing.

Once it is decided that filtering will be applied, one of two filtering processes is chosen depending on the block size and the continuity of the reference samples. By default, a three-tap [1 2 1]/4 smoothing filter is used. The outmost reference samples, $p[-1][2N-1]$ and $p[2N-1][-1]$ are not modified. All other reference samples are filtered by using two neighboring reference samples as:

$$p\,[-1\,]\,[-1] = (p\,[-1\,]\,[\,0] + 2p\,[-1\,]\,[-1] + p\,[\,0\,]\,[-1] + 2) >> 2 \tag{4.1}$$

$$p\,[-1\,]\,[\,y] = (p\,[-1\,]\,[\,y+1] + 2p\,[-1\,]\,[\,y] + p\,[-1\,]\,[\,y-1] + 2) >> 2 \tag{4.2}$$

$$p\,[\,x\,]\,[-1] = (p\,[\,x+1\,]\,[-1] + 2p\,[\,x\,]\,[-1] + p\,[\,x-1\,]\,[-1] + 2) >> 2 \tag{4.3}$$

for $y = 0 \ldots 2N-2$ and $x = 0 \ldots 2N-2$, where ">>" indicates an arithmetic right shift operation.

Otherwise, if the prediction block size is equal to 32×32 and the reference samples are found to be sufficiently flat, the reference samples are generated by applying linear interpolation between the three corner reference samples, $p[-1][63]$, $p[-1][-1]$ and $p[63][-1]$ instead of the three-tap smoothing filter. The flatness is determined by using the following two inequalities based on the experimentally determined thresholds.

$$|\,p\,[-1\,]\,[-1] + p\,[\,2N-1\,]\,[-1] - 2p\,[\,N-1\,]\,[-1]| < (1 << (b-5\,)\,) \tag{4.4}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00106

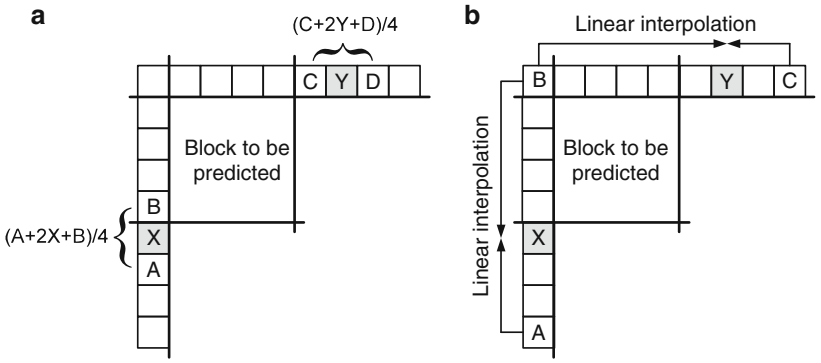


Fig. 4.4 Two types of filtering process of reference samples: (**a**) shows a three-tap filtering using two neighboring reference samples. Reference sample *X* is replaced by the filtered value using *A*, *X* and *B* while reference sample *Y* is replaced by the filtered value using *C*, *Y* and *D*. (**b**) Shows a strong intra smoothing process using corner reference samples. Reference sample *X* is replaced by a linearly filtered value using *A* and *B* while *Y* is replaced by a linearly filtered value using *B* and *C*

$$| \, p\,[-1\,]\,[-1] + p\,[-1\,]\,[\,2N-1] - 2p\,[-1\,]\,[\,N-1]| < (1 << (b-5\,)\,) \quad (4.5)$$

where b specifies the sample bit depth. If the above two inequalities are satisfied and the prediction block size is equal to 32×32, the non-corner reference samples of $p[-1][0], \ldots, p[-1][62]$ and $p[0][-1], \ldots, p[62][-1]$ are generated as:

$$p\,[-1\,]\,[\,y] = ((63-y) * p\,[-1\,]\,[-1] + (y+1) * p\,[-1\,]\,[\,63] + 32) >> 6 \quad (4.6)$$

$$p\,[\,x\,]\,[-1] = ((63-x) * p\,[-1\,]\,[-1] + (x+1) * p\,[\,63\,]\,[-1] + 32) >> 6 \quad (4.7)$$

for $y = 0 \ldots 62$ and $x = 0 \ldots 62$.

This process is referred to as strong intra smoothing as it substitutes nearly all the original reference samples with interpolated ones. The process was developed to remove some blocking and contouring artifacts visible on extremely smooth image areas and it can be selectively turned on or off by the syntax element *strong_intra_smoothing_enabled_flag* in a sequence parameter set. Figure 4.4 illustrates the two types of filtering process of reference samples.

4.3 Intra Sample Prediction

In order to predict different kinds of content efficiently HEVC supports a range of sample prediction methods. The angular intra prediction is designed to model different directional structures typically present in pictures. Whereas planar and DC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00107

prediction modes provide predictions for image areas with smooth and gradually changing content. Planar and DC predictions are also useful in creating "neutral" prediction blocks with no high frequency components for complex textures that cannot be properly modeled with any of the directional predictors that the angular intra prediction is able to generate. In order to further improve the prediction quality, some of the angular modes and the DC prediction mode include a light post-filtering operation to enhance continuity of the prediction signal at block boundaries.

4.3.1 Angular Prediction

Angular intra prediction in HEVC is designed to efficiently model different directional structures typically present in video and image content. The set of available prediction directions has been selected to provide a good trade-off between encoding complexity and coding efficiency for typical video material. The sample prediction process itself is designed to have low computational requirements and to be consistent across different block sizes and prediction directions. This has been found especially important as the number of block sizes and prediction directions supported by HEVC intra coding far exceeds those of previous video codecs, such as H.264/AVC. In HEVC there are four effective intra prediction block sizes ranging from 4×4 to 32×32 samples, each of which supports 33 distinct prediction directions. A decoder must thus support 132 combinations of block size and prediction direction.

4.3.1.1 Angle Definitions

HEVC defines a set of 33 angular prediction directions at 1/32 sample accuracy as illustrated in Fig. 4.5. In natural imagery, horizontal and vertical patterns typically occur more frequently than patterns with other directionalities. Small differences for displacement parameters for modes close to horizontal and vertical directions take advantage of that phenomenon and provide more accurate prediction for nearly horizontal and vertical patterns [10]. The displacement parameter differences become larger closer to diagonal directions to reduce the density of prediction modes for less frequently occurring patterns.

Table 4.2 provides the exact mapping from indicated intra prediction mode to angular parameter A. That parameter defines the angularity of the selected prediction mode (how many 1/32 sample grid units each row of samples is displaced with respect to the previous row).

4.3.1.2 Reference Row Extension for the Negative Prediction Directions

In order to simplify the sample prediction process, the reference samples above the block $p[x][-1]$ and left of the block $p[-1][y]$ are placed in a one dimensional (1-D)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00108

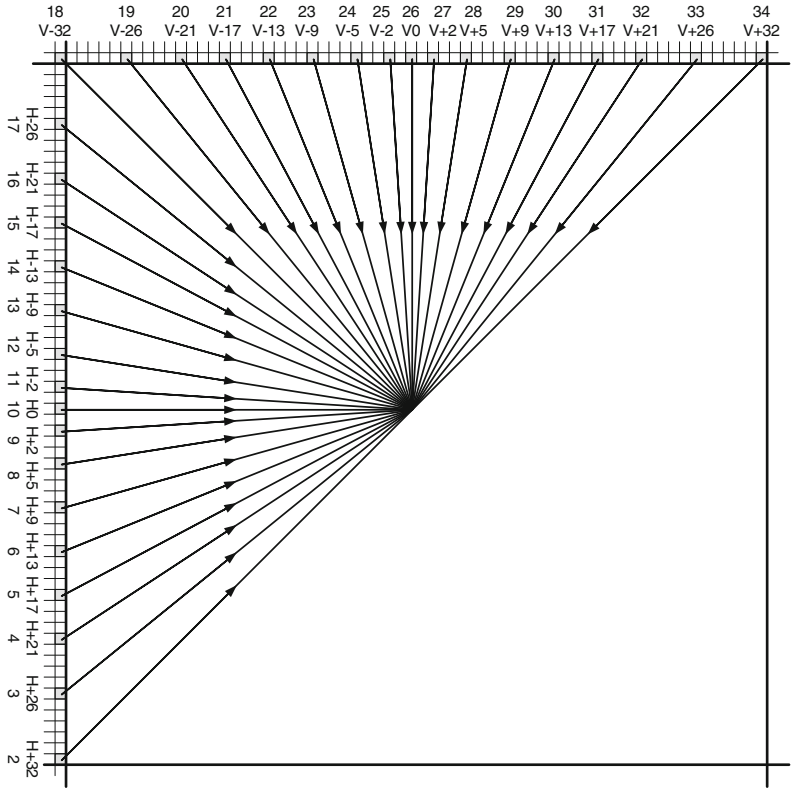


Fig. 4.5 Angle definitions of angular intra prediction in HEVC numbered from 2 to 34 and the associated displacement parameters. *H* and *V* are used to indicate the horizontal and vertical directionalities, respectively, while the numeric part of the identifier refers to the sample position displacements in 1/32 fractions of sample grid positions. Reproduced with permission from [8], © 2012 IEEE

Table 4.2 Angular parameter *A* defines the directionality of each angular intra prediction mode

Horizontal modes																
Mode	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	32	26	21	17	13	9	5	2	0	−2	−5	−9	−13	−17	−21	−26
Vertical modes																
Mode	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
A	−32	−26	−21	−17	−13	−9	−5	−2	0	2	5	9	13	17	21	26

array. For the modes with positive angular parameter *A* (*vertical modes 26 to 33 and horizontal modes 2 to 10*) the 1-D reference array is simply created by copying reference samples from the direction of the prediction:

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00109

Table 4.3 Inverse angle parameter B as a function of angular parameter A

A	-32	-26	-21	-17	-13	-9	-5	-2
B	-256	-315	-390	-482	-630	-910	-1638	-4096

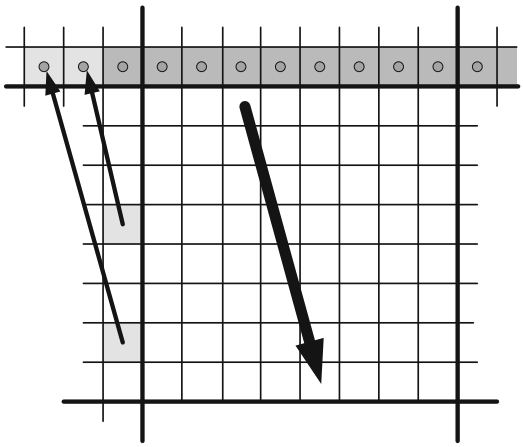


Fig. 4.6 An example of projecting left reference samples to extend the top reference row. The *bold arrow* represents the prediction direction, the *thin arrows* the reference sample projections in the case of intra mode 23 (vertical prediction with a displacement of $-9/32$ samples per row). Reproduced with permission from [8], © 2012 IEEE

$$ref\,[\,x] = p\,[-1 + x\,]\,[-1]\,,(x \geq 0)\ \text{for vertical modes} \tag{4.8}$$

$$ref\,[\,y] = p\,[-1\,]\,[-1 + y]\,,(y \geq 0)\ \text{for horizontal modes} \tag{4.9}$$

If angular parameter A is negative for the selected mode, samples from both left and above of the block are required for the reference array. In these cases the samples in $ref[x]$ array with non-negative indexes are populated as described above. The sample values for negative indexes are obtained by projecting the left reference column to extend the top reference row towards left, or projecting the top reference row to extend the left reference column upwards in the case of vertical and horizontal predictions, respectively. This projection is defined as follows:

$$ref\,[\,x] = p\,[-1\,]\,[-1 + ((x * B + 128) >> 8)]\,,(x < 0)\ \text{for vertical modes} \tag{4.10}$$

$$ref\,[\,y] = p\,[-1 + ((y * B + 128) >> 8\,)\,]\,[-1]\,,(y < 0)\ \text{for horizontal modes} \tag{4.11}$$

where B represents the inverse angle of the angular parameter A and is given in Table 4.3.

Figure 4.6 illustrates extension of the reference row for the vertical prediction mode 23 with angular parameter A equal to -9. In this case, two of the samples in the left reference column are required in the sample prediction process and projected to extend the reference array with inverse angle parameter of -910.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00110

4.3.1.3 Sample Prediction for Angular Prediction Modes

Predicted sample values $p[x][y]$ are obtained by projecting the location of the sample $p[x][y]$ to the reference sample array applying the selected prediction direction and interpolating a value for the sample at 1/32 sample position accuracy. Interpolation is performed linearly utilizing the two closest reference samples in the direction of prediction.

Sample prediction for horizontal modes (modes 2–17) is given by:

$$p[x][y] = ((32 - f) * ref[y + i + 1] + f * ref[y + i + 2] + 16) >> 5 \tag{4.12}$$

And sample prediction for vertical modes (modes 18–34) is given by:

$$p[x][y] = ((32 - f) * ref[x + i + 1] + f * ref[x + i + 2] + 16) >> 5 \tag{4.13}$$

where i is the projected integer displacement on row y (for vertical modes) or column x (for horizontal modes) and calculated as a function of angular parameter A as follows:

$$i = ((x + 1) * A) >> 5, \text{for horizontal modes} \tag{4.14}$$

$$i = ((y + 1) * A) >> 5, \text{for vertical modes} \tag{4.15}$$

f represents the fractional part of the projected displacement on the same row or column and is calculated as:

$$f = ((x + 1) * A)\,\&31, \text{for horizontal modes} \tag{4.16}$$

$$f = ((y + 1) * A)\,\&31, \text{for vertical modes} \tag{4.17}$$

The projection parameters i and f are constant for a row of samples in vertical prediction and a column of samples in horizontal predictions. Thus, the only operation that needs to be performed for each sample is the linear interpolation to derive $p[x][y]$. It should be also noted that in the case the fractional parameter f is zero, the interpolation can be omitted and a sample from the reference array can be directly used as the predicted sample.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00111

4.3.2 DC Prediction

In the case of DC prediction, the predicted sample values are populated with a constant value representing the average of the reference samples immediately left and to the above of the block to be predicted. The DC predicted luminance blocks of size 16×16 and smaller go through a light filtering process to soften the left and above edges of the block as described in Sect. 4.3.4.

4.3.3 Planar Prediction

While angular prediction provides good approximations for structures with significant edges, it can create some visible contouring in smooth image areas. Similarly, some blockiness can typically be observed in smooth image areas when DC prediction is applied at low or medium bit rates. The planar prediction of HEVC is designed to overcome these issues by having the capability to generate a prediction surface without discontinuities on the block boundaries. This is achieved by averaging a horizontal and vertical linear prediction on sample basis as follows:

$$p[x][y] = (p_h[x][y] + p_v[x][y] + N) >> (\log_2(N) + 1) \quad (4.18)$$

The horizontal prediction $p_h[x][y]$ and the vertical prediction $p_v[x][y]$ for location $[x][y]$ are calculated as follows:

$$p_h[x][y] = (N - 1 - x) * p[-1][y] + (x + 1) * p[N][-1] \quad (4.19)$$

$$p_v[x][y] = (N - 1 - y) * p[x][-1] + (y + 1) * p[-1][N] \quad (4.20)$$

Figure 4.7 illustrates derivation of predicted sample values in the case of planar mode. The top-right reference sample $p[N][-1]$ is used as the right reference for all the horizontal filtering. Similarly, the bottom-left reference sample $p[-1][N]$ is used as the bottom reference for all the vertical operations. Final prediction value for each sample is obtained by averaging the horizontal and vertical predictions.

4.3.4 Post-processing for Predicted Samples

Some of the prediction modes can generate discontinuities for the predicted sample values on the boundaries of the prediction blocks. Such cases are most evident for the DC prediction and for the directly horizontal and vertical angular predictions. In the case of DC prediction, such discontinuity can occur on both top and left boundary of the block as all the predicted samples are replaced with a single value.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00112

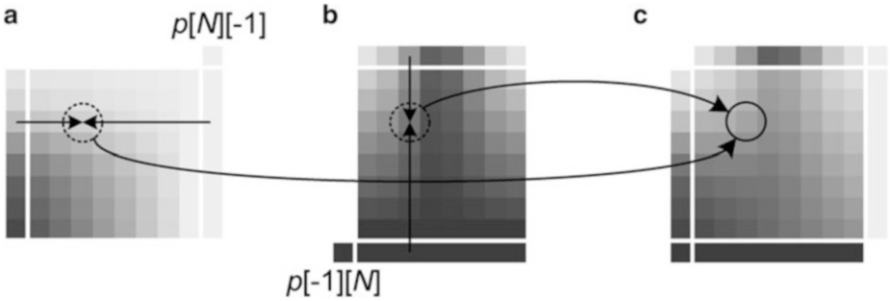


Fig. 4.7 Example of planar prediction: (**a**) illustrates calculation of the horizontal component, (**b**) illustrates generation of the vertical component and (**c**) provides an example of a final planar prediction created by averaging the horizontal and vertical components

In the case of directly vertical prediction, there may be discontinuities on the left boundary of the block as the leftmost column of predicted samples replicate the value of the leftmost reference sample above the block. Similar discontinuities can occur on the top edge of the block when directly horizontal prediction is selected.

To reduce these discontinuities along the block boundaries, the boundary samples inside the prediction block are replaced by the filtered values by considering the slope at the edge of the block [11]. This boundary smoothing is applied only for the abovementioned three intra modes (DC prediction, the exactly horizontal angular mode 26 and the exactly vertical angular mode 10) and when the prediction block size is smaller than 32×32. Different approaches for other modes and block sizes were studied during the course of HEVC development, but this combination of modes and block sizes was found to provide a desirable balance between coding efficiency and complexity. As the prediction for chroma components tends to be very smooth, the benefits of the boundary smoothing would be limited. Thus, in order to avoid extra processing with marginal quality improvements the prediction boundary smoothing is only applied to the luma component.

When the intra mode is equal to exactly vertical direction, the prediction samples $p[0][y]$ with $y = 0 \ldots N-1$ are modified by using the sample difference between two reference samples $p[-1][-1]$ and $p[-1][y]$ as:

$$p[0][y] = p[0][y] + ((p[-1][y] - p[-1][-1]) >> 1) \quad \text{for } y = 0 \ldots N-1 \tag{4.21}$$

where clipping operation to restrict the computed value within the sample bit-depth is omitted for simplicity. For the exactly horizontal direction, the boundary smoothing is done in a similar way.

In the case of DC prediction, a three-tap [1 2 1]/4 smoothing filter is applied to the first prediction sample p[0][0] by using the predicted DC value *dcVal* and two neighboring reference samples, $p[-1][0]$ and $p[0][-1]$, as follows:

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00113

$$p[0][0] = (p[-1][0] + 2 * dcVal + p[0][-1] + 2) >> 2 \quad (4.22)$$

For other boundary samples, a two-tap [3 1]/4 smoothing filter is applied with the higher weight on the DC value:

$$p[x][0] = (p[x][-1] + 3 * dcVal + 2) >> 2 \quad \text{for } x = 1 \ldots N-1 \quad (4.23)$$

$$p[0][y] = (p[-1][y] + 3 * dcVal + 2) >> 2 \quad \text{for } y = 1 \ldots N-1 \quad (4.24)$$

4.4 Intra Mode Coding

While increasing the number of intra prediction modes provides better prediction, efficient intra mode coding is required to ensure that the selected mode is signaled with minimal overhead. For luma component, three most probable modes are derived to predict the intra mode instead of a single most probable one as in H.264/AVC. Possible redundancies among the three most probable modes are also considered and redundant modes are substituted with alternative ones to maximize the signaling efficiency. For chroma intra mode, HEVC introduces a derived mode which allows efficient signaling of the likely scenario where chroma is using the same prediction mode as luma. The syntax elements for signaling luma and chroma intra modes are designed by utilizing the increased number of most probable modes for the luma component and the statistical behavior of the chroma component.

4.4.1 Prediction of Luma Intra Mode

HEVC supports a total of 33 angular prediction modes as well as planar and DC prediction for luma intra prediction for all the PU sizes. Due to the large number of intra prediction modes, H.264/AVC-like mode coding approach based on a single most probable mode was not effective in HEVC. Instead, HEVC defines three most probable modes for each PU based on the modes of the neighboring PUs. The selected number of most probable modes makes it also possible to indicate one of the 32 remaining modes by a CABAC bypassed fixed-length code, as distribution of the mode probabilities outside of the set of most probable modes has been found to be relatively uniform.

The selection of the set of three most probable modes is based on modes of two neighboring PUs, one left and one to the above of the current PU. Let the intra modes of left and above of the current PU be A and B, respectively. If a neighboring PU is not coded as intra or is coded with pulse code modulation (PCM) mode, the PU is considered to be a DC predicted one. In addition, B is assumed to be DC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00114

mode when the above neighboring PU is outside the CTU to avoid introduction of an additional line buffer for intra mode reconstruction.

If A is not equal to B, the first two most probable modes denoted as $MPM[0]$ and $MPM[1]$ are set equal to A and B, respectively, and the third most probable mode denoted as $MPM[2]$ is determined as follows:

- If neither of A or B is planar mode, $MPM[2]$ is set to planar mode.
- Otherwise, if neither of A or B is DC mode, $MPM[2]$ is set to DC mode.
- Otherwise (one of the two most probable modes is planar and the other is DC), $MPM[2]$ is set equal to angular mode 26 (directly vertical).

If A is equal to B, the three most probable modes are determined as follows. In the case they are not angular modes (A and B are less than 2), the three most probable modes are set equal to planar mode, DC mode and angular mode 26, respectively. Otherwise (A and B are greater than or equal to 2), the first most probable mode $MPM[0]$ is set equal to A and two remaining most probable modes $MPM[1]$ and $MPM[2]$ are set equal to the neighboring directions of A and calculated as:

$$MPM\,[\,1] = 2 + ((A - 2 - 1 + 32)\,\%\;32\,) \tag{4.25}$$

$$MPM\,[\,2] = 2 + ((A - 2 + 1)\,\%\;32\,) \tag{4.26}$$

where % denotes the modulo operator (i.e., $a\,\%\,b$ denotes the remainder of a divided by b). Figure 4.8 summarizes the derivation process for the three most probable modes $MPM[0]$, $MPM[1]$ and $MPM[2]$ from the neighboring intra modes A and B.

The three most probable modes $MPM[0]$, $MPM[1]$ and $MPM[2]$ identified as described above are further sorted in an ascending order according to their mode number to form the ordered set of most probable modes. If the current intra prediction mode is equal to one of the elements in the set of most probable modes, only the index in the set is transmitted to the decoder. Otherwise, a 5-bit CABAC bypassed codeword is used to specify the selected mode outside of the set of most probable modes as the number of modes outside of the set is equal to 32. The bitstream syntax design for signaling intra modes is described in Sect. 4.4.3.

4.4.2 Derived Mode for Chroma Intra Prediction

To enable the use of the increased number of directionalities in the chroma intra prediction while minimizing the signaling overhead, HEVC introduces the INTRA_DERIVED mode to indicate the cases when a chroma PU uses the same prediction mode as the corresponding luma PU. More specifically, for a chroma PU one of the five chroma intra prediction modes: planar, angular 26 (directly vertical), angular 10 (directly horizontal), DC or derived mode is signaled.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00115

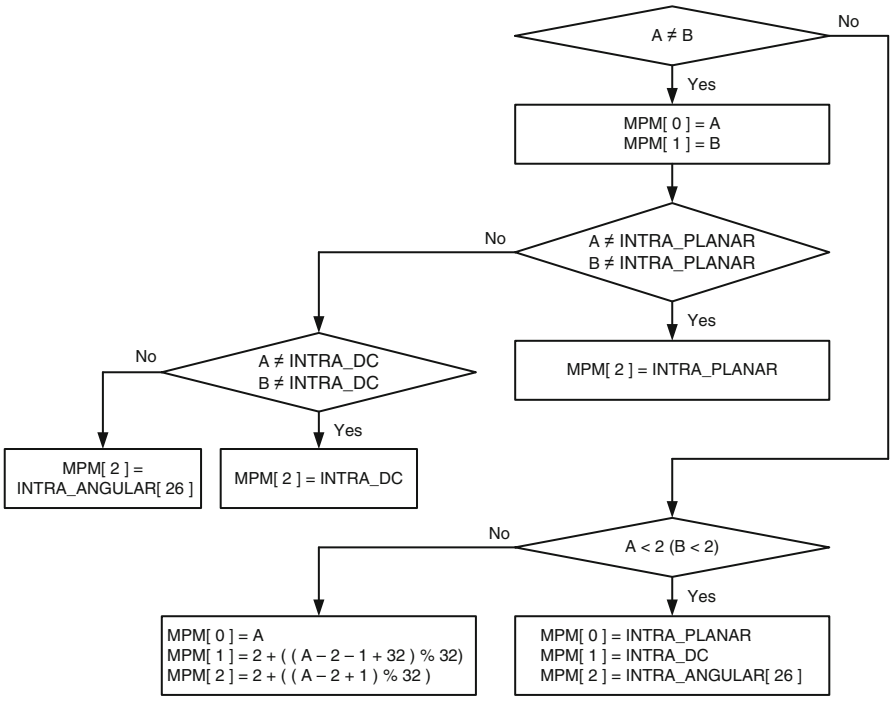


Fig. 4.8 Derivation process for the three most probable modes *MPM*[0], *MPM*[1] and *MPM*[2]. *A* and *B* indicate the neighboring intra modes of the left and the above PU, respectively

Table 4.4 Determination of chroma intra prediction mode according to luma intra prediction mode

Initial chroma intra mode	Final chroma intra mode when derived mode ≠ initial chroma intra mode	Final chroma intra mode when derived mode = initial chroma intra mode
INTRA_PLANAR	INTRA_PLANAR	INTRA_ANGULAR[34]
INTRA_ANGULAR[26]	INTRA_ANGULAR[26]	INTRA_ANGULAR[34]
INTRA_ANGULAR[10]	INTRA_ANGULAR[10]	INTRA_ANGULAR[34]
INTRA_DC	INTRA_DC	INTRA_ANGULAR[34]
INTRA_DERIVED	Luma intra mode	N/A

This design is based on the finding that often structures in the chroma signal follow those of the luma. In the case the derived mode is indicated for a chroma PU, intra prediction is performed by using the corresponding luma PU mode. Angular intra mode 34 is used to substitute the four chroma prediction modes with individual identifiers when the derived mode is one of those four modes. The substitution process is illustrated in Table 4.4.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00116

Table 4.5 Binarization of the syntax element `intra_chroma_pred_mode` with respect to the chroma intra prediction mode

Chroma intra prediction mode	Binarization of `intra_chroma_pred_mode`
INTRA_DERIVED	0
INTRA_PLANAR	100
INTRA_ANGULAR[26]	101
INTRA_ANGULAR[10]	110
INTRA_DC	111

4.4.3 Syntax Design for Intra Mode Coding

Let three sorted most probable modes in an ascending order for luma intra mode be *SMPM*[0], *SMPM*[1] and *SMPM*[2], respectively. The syntax element `prev_intra_luma_pred_flag` specifies whether the luma intra mode is equal to one of the three most probable modes. In this case, the additional syntax element `mpm_idx` indicates that *SMPM*[`mpm_idx`] is the selected luma intra mode.

Otherwise (`prev_intra_luma_pred_flag` is equal to 0), the syntax element `rem_intra_luma_pred_mode` is used to specify the luma intra mode. Since the total number of luma intra modes is equal to 35 and the number of most probable modes is equal to 3, the range of `rem_intra_luma_pred_mode` is from 0 to 31, inclusive, which can be expressed by a 5-bit codeword conveniently. Given `rem_intra_luma_pred_mode`, the luma intra mode is derived as follows:

1. Let L be equal to the value of `rem_intra_luma_pred_mode`.
2. For $i = 0 \ldots 2$ (sequentially in an increasing order), L is incremented by one if L is greater than or equal to *SMPM*[i].

For chroma intra mode coding, it has been observed that the derived mode is used most frequently while other four chroma intra modes with their own identifiers (planar, angular 26, angular 10 and DC modes) are typically used with roughly similar probabilities. The syntax element `intra_chroma_pred_mode` is binarized with respect to the above observations as given in Table 4.5.

4.5 Encoding Algorithms

Due to the large number of intra modes in HEVC, the computations of the rate-distortion costs for all intra modes are impractical for most of the applications. In the HEVC reference software, the sum of absolute transformed differences (SATD) between prediction and original samples is used to reduce the number of luma intra mode candidates before applying rate-distortion optimization (RDO) [12].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00117

The number of luma intra mode candidates entering full RDO is determined according to the corresponding PU sizes as eight for PUs of size 4×4 and 8×8 PU, and three for other PU sizes. For those luma intra mode candidates as well as luma intra modes which are a part of the most probable mode set, intra sample prediction and transform coding are performed to obtain both the number of required bits and the resulting distortion. Finally, the luma intra mode which minimizes the rate-distortion cost is selected. For the chroma intra encoding, all possible intra chroma modes are evaluated based on their rate-distortion costs as the number of intra chroma modes is much smaller than that of luma modes. It has been reported that this kind of technique can provide negligible coding efficiency loss (less than 1 % increase in bit rate at aligned objective quality) compared to a full rate-distortion search while reducing the encoding complexity by a factor of three [8].

In the literature, several fast intra coding algorithms have been studied in the context of HEVC. One popular approach for achieving this purpose is to reduce the number of block sizes to be tested in the HEVC intra coding. Shen et al. [13] proposed a fast CU size decision and mode decision algorithm for HEVC intra coding based on the previous decisions in spatially nearby blocks. It results to about 21 % encoding time decrease with negligible loss of coding efficiency. In [4], the fast CU splitting and pruning decisions according to a Bayes decision rule for the HEVC intra coding was proposed and about 50 % encoding time reduction was reported with 0.6 % bit rate increases.

Another approach is to reduce the number of intra mode candidates based on local image characteristics. Zhao et al. [19] proposed forced use of the most probable mode with a reduced number of luma intra mode candidates in the rate-distortion optimization stage. Around 20 % encoding time reduction was achieved with this approach and the method was also decided to be included in the HEVC reference software. In [3, 5, 14], it has been shown that reducing the intra mode candidates based on directionalities of the source and neighboring blocks about 20–30 % encoding time reduction can be achieved.

4.6 Coding Efficiency and Decoder Complexity

4.6.1 Coding Efficiency

Coding efficiency of HEVC intra coding has been reported to exceed significantly that of earlier video and still picture codecs. For example, Hanhart et al. [6] evaluated both objective (PSNR) and subjective (MOS) quality of HEVC intra coding, JPEG and JPEG 2000 using the JPEG XR test set. Test material in this case consisted solely of camera captured photographic content. The study concluded that HEVC outperformed legacy still picture codecs with most significant differences at bit rates below 1.00 bpp. Table 4.6 summarizes the average bit rate reductions HEVC obtained over other codecs in this study.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00118

Table 4.6 Average objective and subjective bit rate reductions provided by HEVC with respect to JPEG and JPEG 2000 as reported in [6]

Measure	JPEG (%)	JPEG 2000 (%)
ΔBR (PSNR)	−61.6	−20.3
ΔBR (MOS)	−43.1	−31.0

Table 4.7 Average objective bit rate reductions provided by HEVC with respect to JPEG, JPEG XR and JPEG 2000 as reported in [16]

Quality range	JPEG (%)	JPEG XR (%)	JPEG 2000 (%)
High	−49.3	−32.2	−26.0
Medium	−58.2	−39.8	−30.7
Low	−62.5	−42.6	−26.1

Similar levels of improvement over legacy codecs were reported in [16] using the JCT-VC test set and Bjøntegaard delta metrics [1]. This test set includes material mostly captured with video cameras, but it also contains some computer generated material as well as typical screen content. Table 4.7 summarizes the results of this study which analyzed performance at three different quality ranges: high (HEVC QPs 12–27), medium (HEVC QPs 22–37) and low (HEVC QPs 32–47). In these tests corresponding reference implementations were used and the resulting PSNRs for each codec were aligned by adjusting the respective quality factors.

Coding efficiency improvements over H.264/AVC intra coding has been analyzed for example in [9]. Results of that study indicate that HEVC requires on average 23 % less bit rate than H.264/AVC for still picture coding. The report also points out that the subjective differences may be larger than that as some of the new HEVC tools are expected to contribute more to the visual image quality than to the objective performance of the codec.

4.6.2 Decoder Complexity

Although estimating the decoding complexity accurately can be an extremely difficult task due to various architectural aspects depending on the implementation platform, analyzing the traditional operational cycle counts can give reasonable estimates about relative complexities of different algorithms.

The DC prediction, the exactly horizontal angular mode 26 and the exactly vertical angular mode 10 in the HEVC are similar to those specified in the H.264/AVC. Although the additional boundary smoothing operation is required in HEVC, its impact becomes negligible when the prediction block size becomes larger due to the smaller relative area of the filtered samples.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00119

The elementary operation of the angular prediction described in Sect. 4.3.1.3 can be simplified as:

$$p = (u * a + v * b + 16) >> 5 \tag{4.27}$$

where u and v are two pre-computed constants, and a and b are two pre-fetched reference sample values. It suggests that the angular prediction requires a total of five operations including two multiplications, two additions and one shift per one predicted sample. In the H.264/AVC case, the directional prediction process can be expressed as:

$$p = (a + 2 * b + c + 2) >> 2 \tag{4.28}$$

where a, b and c are three pre-fetched reference sample values. It means that the H.264/AVC process requires also a total of five operations including three additions and two shifts.

The directional prediction of H.264/AVC can be implemented by using only multiplication-free operations easily, thus it is reported to have less computational complexity compared with the angular prediction of HEVC in some specific architecture, but the difference tends to be relatively minor [8]. On the other hand, all the HEVC angular prediction modes use the same prediction process regardless of the block sizes whereas H.264/AVC defines slightly different operations for 4×4, 8×8 and 16×16 block sizes. This property of HEVC can be considered very useful when it comes to implementing the HEVC prediction modes as there is such a large amount of different combinations of prediction block sizes and the prediction directions in HEVC.

In [8], a more detailed decoder complexity analysis has been carried out by using decoding times measured in an optimized software implementation [2]. For the detailed information such as the relative decoding time of intra prediction to overall decoding time or cycle counts for each prediction mode, please refer to [8].

4.7 Main Still Picture Profile and Its Applications

HEVC version 1 includes a specific profile for still picture coding purposes [15]. The "Main Still Picture profile" defines bitstream characteristics and decoder operation for using the HEVC toolset to efficiently represent both typical camera captured still pictures as well as synthetic images. Typical use cases for the profile include traditional photography, capturing still pictures during video recording, extraction of still pictures from video sequences, sharing pictures over Internet or other communication channels, and different computational photography applications. HEVC Still Picture profile is able to provide a very bandwidth efficient way to represent high quality images in each of these use cases. In the future extensions

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00120

of the standard it is expected that the HEVC still imaging features will be further extended towards 4:4:4 color formats and high dynamic range photography requiring support for bit-depths above 8 bits per sample.

The key characteristics of the HEVC Main Still Picture profile can be summarized as follows:

- HEVC Main profile intra coding tools are used
  - Chroma format is limited to 4:2:0
  - Bit depth is limited to 8 bits per sample
  - Coding tree block size can vary from 16×16 to 64×64
- Bitstream contains only one picture
- Decoded picture buffer is of size one picture

4.8 Summary of Differences from H.264/AVC

Both HEVC intra coding and H.264/AVC intra coding are based on spatial sample prediction followed by transform coding [18]. However, the intra coding methods in HEVC are more elaborated in number of ways. Firstly, the set of supported prediction block sizes is extended up to 32×32 to be aligned with the HEVC coding structures and to improve reconstruction of smooth image areas. Secondly, the number of available directional modes is extended from 8 to 33 to improve modelling of directional textures. In HEVC the whole range of directional predictors is made available for both luma and chroma blocks, while in H.264/AVC the number of available directional prediction modes for chroma is limited to two (namely the directly horizontal and vertical modes). Also sample continuities along the block boundaries are considered more by redefining the planar mode and applying post-processing for directly horizontal, vertical and the DC mode. Another advantage of HEVC is its ability to pad the missing reference samples and allow usage of all the prediction modes independent of availability of certain reference samples. The HEVC intra mode coding also uses an approach different from that of H.264/AVC. Due to the large number of intra modes in HEVC, the luma intra mode is signaled by using three most probable modes and the selected luma mode is always made available as one of the candidate modes for the corresponding chroma blocks. Table 4.8 summarizes key differences of intra prediction in H.264/AVC and HEVC. Lainema et al. [8] provides further analysis on contribution of individual intra prediction tools to the overall coding efficiency.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00121

Table 4.8 Key differences of intra prediction techniques between H.264/AVC and HEVC

Functionality	H.264/AVC	HEVC
Prediction block sizes	4×4, 8×8 and 16×16	4×4, 8×8, 16×16 and 32×32
Luma intra prediction modes	9 (4×4 and 8×8), 4 (16×16)	35
Chroma intra prediction modes	4	4 + the luma mode
Reference sample smoothing	8×8	8×8 and above
Boundary smoothing	N/A	Used for directly horizontal, vertical and DC modes
Operation when reference samples missing	Use DC mode	Reference sample substitution
Number of most probable modes in mode coding	1	3

References

1. Bjøntegaard G (2008) Improvements of the BD-PSNR model, ITU-T SG16/Q6 Video Coding Experts Group (VCEG), Document VCEG-AI11, Berlin, July 2008
2. Bossen F (2011) On Software Complexity, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G757, Geneva, Nov. 2011
3. Chen G, Liu Z, Ikenaga T, Wang D (2013) Fast HEVC intra mode decision using matching edge detector and kernel density estimation alike histogram generation. In: Proceedings of IEEE international symposium on circuits and systems (ISCAS), May 2013, pp 53–56
4. Cho S, Kim M (2013) Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding. IEEE Trans Circuits Syst Video Technol 23(9):1555–1564
5. Jiang W, Ma H, Chen Y (2012) Gradient based fast mode decision algorithm for intra prediction in HEVC. In: Proceedings of international conference on Consumer Electronics, Communications and Networks (CECNet), Apr. 2012, pp 1836–1840
6. Hanhart P, Rerabek M, Korshunov P, Ebrahimi T (2013) Subjective evaluation of HEVC intra coding for still image compression, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L0380, Geneva, Jan. 2013
7. Kanumuri S, Tan TK, Bossen F (2011) Enhancements to intra coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D235, Daegu, Jan. 2011
8. Lainema J, Bossen F, Han W-J, Min J, Ugur K (2012) Intra coding of the HEVC standard. IEEE Trans Circuits Syst Video Technol 22(12):1792–1801
9. Li B, Sullivan GJ, Xu J (2013) Comparison of compression performance of HEVC draft 10 with AVC high profile, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-M0041, Incheon, Apr. 2013
10. Min J, Lee S, Kim I, Han W-J, Lainema J, Ugur K (2010) Unification of the directional intra prediction methods in TMuC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-B100, Geneva, July 2010
11. Minezawa A, Sugimoto K, Sekiguchi S (2011) An improved intra vertical and horizontal prediction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F172, Torino, July 2011
12. Piao Y, Min J, Chen J (2010) Encoder improvement of unified intra prediction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C207, Guangzhou, Oct. 2010
13. Shen L, Zhang Z, An P (2013) Fast CU size decision and mode decision algorithm for HEVC intra coding. IEEE Trans Consum Electron 59(1):207–213

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00122

14. Silva TL, Agostini LV, Silva Cruz LA (2012) Fast HEVC intra prediction mode decision based on EDGE direction information. In: Proceedings of 20th European signal processing conference (EUSIPCO), Aug. 2012, pp 1214–1218
15. Sullivan GJ, Ohm J-R, Han W-J, Wiegand T (2012) Overview of high efficiency video coding (HEVC) standard. IEEE Trans Circuits Syst Video Technol 22(12):1649–1668
16. Ugur K, Lainema J (2013) Updated results on HEVC still picture coding performance, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-M0041, Incheon, Apr. 2013
17. Ugur K, Andersson KR, Fuldseth A (2010) Video coding technology proposal by Tandberg, Nokia, and Ericsson, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A119, Dresden, Apr. 2010
18. Wiegand T, Sullivan GJ, Bjøntegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576
19. Zhao L, Zhang L, Ma S, Zhao D (2011) Fast mode decision algorithm for intra prediction in HEVC. In: Proceedings of visual communications and image processing (VCIP), Nov. 2011, O-02-4, pp 1–4

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00123

Chapter 5
Inter-Picture Prediction in HEVC

Benjamin Bross, Philipp Helle, Haricharan Lakshman, and Kemal Ugur

Abstract Inter-picture prediction in HEVC can be seen as a steady improvement and generalization of all parts known from previous video coding standards, e.g. H.264/AVC. The motion vector prediction was enhanced with advanced motion vector prediction based on motion vector competition. An inter-prediction block merging technique significantly simplified the block-wise motion data signaling by inferring all motion data from already decoded blocks. When it comes to interpolation of fractional reference picture samples, high precision interpolation filter kernels with extended support, i.e. 7/8-tap filter kernels for luma and 4-tap filter kernels for chroma, improve the filtering especially in the high frequency range. Finally, the weighted prediction signaling was simplified by either applying explicitly signaled weights for each motion compensated prediction or just averaging two motion compensated predictions. This chapter provides a detailed description of these aspects of HEVC standard and explains their coding efficiency and complexity characteristics.

5.1 Introduction

In HEVC, the same basic hybrid video coding approach as in previous standards is applied. Hybrid video coding is known to be a combination of video sample prediction and transformation of the prediction error, i.e. the residual, followed by entropy coding of the prediction information and the transform coefficients.

B. Bross (✉) • P. Helle • H. Lakshman
Fraunhofer Institute for Telecommunications - Heinrich Hertz Institute,
Einsteinufer 37, D-10587 Berlin, Germany
e-mail: benjamin.bross@hhi.fraunhofer.de

K. Ugur
Nokia Corporation, Tampere 33720, Finland
e-mail: Kemal.Ugur@nokia.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_5
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00124

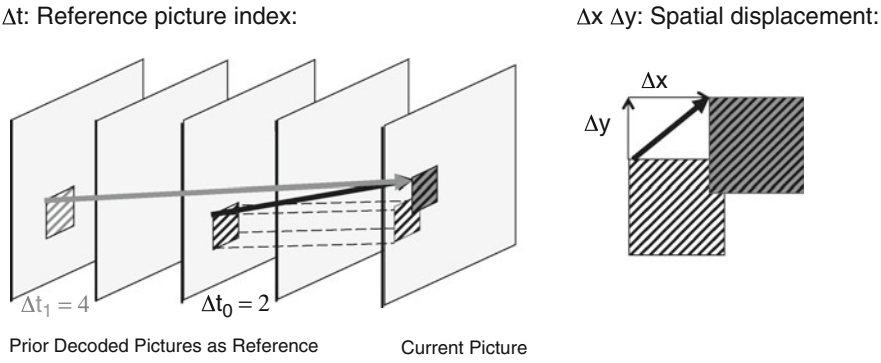


Fig. 5.1 Inter-picture prediction concept and parameters using a translational motion model

While intra-picture prediction exploits the correlation between spatially neighboring samples, inter-picture prediction makes use of the temporal correlation between pictures in order to derive a motion-compensated prediction (MCP) for a block of image samples.

For this block-based MCP, a video picture is divided into rectangular blocks. Assuming homogeneous motion inside one block and that moving objects are larger than one block, for each block, a corresponding block in a previously decoded picture can be found that serves as a predictor. The general concept of MCP based on a translational motion model is illustrated in Fig. 5.1. Using a translational motion model, the position of the block in a previously decoded picture is indicated by a motion vector $(\Delta x, \Delta y)$ where Δx specifies the horizontal and Δy the vertical displacement relative to the position of the current block. The motion vectors $(\Delta x, \Delta y)$ could be of fractional sample accuracy to more accurately capture the movement of the underlying object. Interpolation is applied on the reference pictures to derive the prediction signal when the corresponding motion vector has fractional sample accuracy. The previously decoded picture is referred to as the reference picture and indicated by a reference index Δt to a reference picture list. These translational motion model parameters, i.e. motion vectors and reference indices, are further referred to as motion data. Two kinds of inter-picture prediction are allowed in modern video coding standards, namely uni-prediction and bi-prediction.

In case of bi-prediction, two sets of motion data $(\Delta x_0, \Delta y_0, \Delta t_0$ and $\Delta x_1, \Delta y_1, \Delta t_1)$ are used to generate two MCPs (possibly from different pictures), which are then combined to get the final MCP. Per default, this is done by averaging but in case of weighted prediction, different weights can be applied to each MCP, e.g. to compensate for scene fade outs. The reference pictures that can be used in bi-prediction are stored in two separate lists, namely list 0 and list 1. In order to limit the memory bandwidth in slices allowing bi-prediction, the HEVC standard restricts PUs with 4×8 and 8×4 luma prediction blocks to use uni-prediction only. Motion data is derived at the encoder using a motion estimation process. Motion

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00125

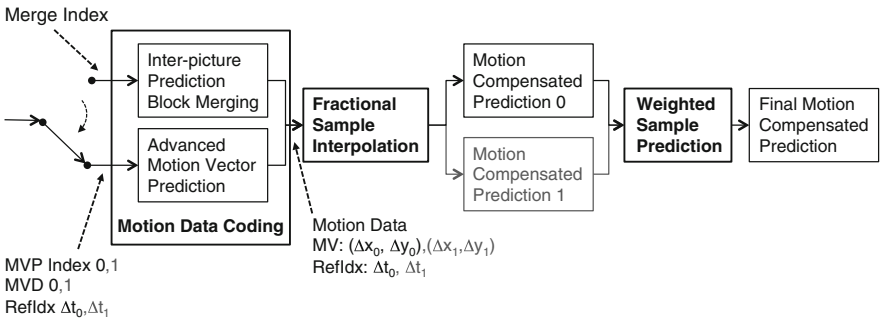


Fig. 5.2 Inter-picture prediction in HEVC (*grey parts* represent the bi-prediction path)

estimation is not specified within video standards so different encoders can utilize different complexity-quality tradeoffs in their implementations.

An overview block diagram of the HEVC inter-picture prediction is shown in Fig. 5.2. The motion data of a block is correlated with the neighboring blocks. To exploit this correlation, motion data is not directly coded in the bitstream but predictively coded based on neighboring motion data. In HEVC, two concepts are used for that. The predictive coding of the motion vectors was improved in HEVC by introducing a new tool called advanced motion vector prediction (AMVP) where the best predictor for each motion block is signaled to the decoder. In addition, a new technique called inter-prediction block merging derives all motion data of a block from the neighboring blocks replacing the direct and skip modes in H.264/AVC [26]. Section 5.2 describes all aspects of motion data coding in HEVC including AMVP, inter-prediction block merging and motion data storage reduction. The improved fractional sample interpolation filter is explained in Sect. 5.3. Additional weighting of the MCP or, in case of bi-prediction, the weighting of the two MCPs is further detailed in Sect. 5.4. Finally, Sect. 5.5 summarizes and concludes this chapter.

5.2 Motion Data Coding

5.2.1 Advanced Motion Vector Prediction

As in previous video coding standards, the HEVC motion vectors are coded in terms of horizontal (x) and vertical (y) components as a difference to a so called motion vector predictor (MVP). The calculation of both motion vector difference (MVD) components is shown in Eqs. (5.1) and (5.2).

$$\mathrm{MVD}_x = \Delta x - \mathrm{MVP}_x \tag{5.1}$$

$$\mathrm{MVD}_y = \Delta y - \mathrm{MVP}_y \tag{5.2}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00126

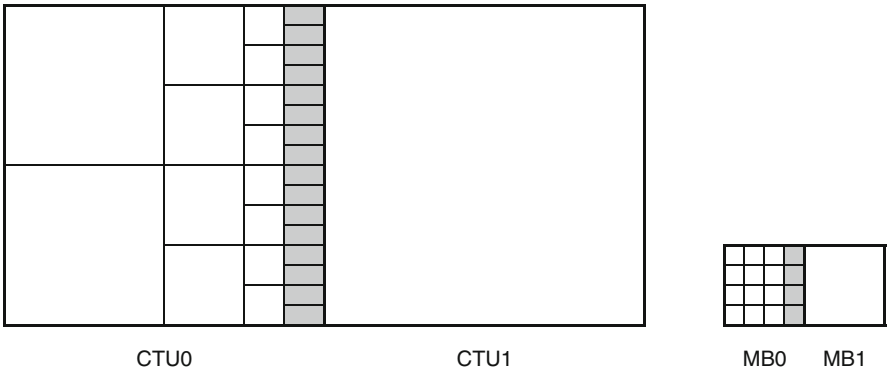


Fig. 5.3 Maximum number of left neighbors with motion data in HEVC with CTU0 having 16 8×4 luma PBs next to CTU1 with one 64×64 luma PB (*left*) and in H.264/AVC with MB0 having four 4×4 partitions next to MB1 with one 16× 16 partition (*right*)

Motion vectors of the current block are usually correlated with the motion vectors of neighboring blocks in the current picture or in the earlier coded pictures. This is because neighboring blocks are likely to correspond to the same moving object with similar motion and the motion of the object is not likely to change abruptly over time. Consequently, using the motion vectors in neighboring blocks as predictors reduces the size of the signaled motion vector difference. The MVPs are usually derived from already decoded motion vectors from spatial neighboring blocks or from temporally neighboring blocks in the co-located picture.[1] In H.264/AVC, this is done by doing a component wise median of three spatially neighboring motion vectors. Using this approach, no signaling of the predictor is required. Temporal MVPs from a co-located picture are only considered in the so called temporal direct mode of H.264/AVC. The H.264/AVC direct modes are also used to derive other motion data than the motion vectors. Hence, they relate more to the block merging concept in HEVC and are further discussed in Sect. 5.2.2.

In HEVC, the approach of implicitly deriving the MVP was replaced by a technique known as motion vector competition, which explicitly signals which MVP from a list of MVPs, is used for motion vector derivation [19]. The variable coding quadtree block structure in HEVC can result in one block having several neighboring blocks with motion vectors as potential MVP candidates. Taking the left neighbor as an example, in the worst case a 64×64 luma prediction block could have 16 8×4 luma prediction blocks to the left when a 64×64 luma coding tree block is not further split and the left one is split to the maximum depth. Figure 5.3 illustrates this example and compares it to the worst case in H.264/AVC. *Advanced Motion Vector Prediction (AMVP)* was introduced to modify motion vector competition to account for such a flexible block structure [11]. During

[1] In some cases, the zero motion vector can also be used as MVP.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00127

the development of HEVC, the initial AMVP design was significantly simplified to provide a good trade-off between coding efficiency and an implementation friendly design. Section 5.2.1.1 describes in detail how the list of potential MVPs is constructed in HEVC. Section 5.2.1.2 describes the signaling of all motion data, including the index to the AMVP list, when AMVP is used for MV coding.

5.2.1.1 AMVP Candidate List Construction

The initial design of AMVP included five MVPs from three different classes of predictors: three motion vectors from spatial neighbors, the median of the three spatial predictors and a scaled motion vector from a co-located, temporally neighboring block. Furthermore, the list of predictors was modified by reordering to place the most probable motion predictor in the first position and by removing redundant candidates to assure minimal signaling overhead. Exhaustive experiments throughout the standardization process investigated how the complexity of this motion vector prediction and signaling scheme could be reduced without sacrificing too much coding efficiency [7, 8, 14]. This led to significant simplifications of the AMVP design such as removing the median predictor, reducing the number of candidates in the list from five to two, fixing the candidate order in the list and reducing the number of redundancy checks. The final design of the AMVP candidate list construction includes the following two MVP candidates:

- up to two *spatial candidate* MVPs that are derived from five spatial neighboring blocks
- one *temporal candidate* MVPs derived from two temporal, co-located blocks when both spatial candidate MVPs are not available or they are identical
- zero motion vectors when the spatial, the temporal or both candidates are not available

Spatial Candidates

As already mentioned, two spatial MVP candidates A and B are derived from five spatially neighboring blocks which are shown in Fig. 5.4b. The locations of the spatial candidate blocks are the same for both AMVP and inter-prediction block merging that will be presented in Sect. 5.2.2.

The derivation process flow for the two spatial candidates A and B is depicted in Fig. 5.5. For candidate A, motion data from the two blocks A0 and A1 at the bottom left corner is taken into account in a two pass approach. In the first pass, it is checked whether any of the candidate blocks contain a reference index that is equal to the reference index of the current block. The first motion vector found will be taken as candidate A. When all reference indices from A0 and A1 are pointing to a different reference picture than the reference index of the current block, the associated motion vector cannot be used as is. Therefore, in a second pass, the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00128

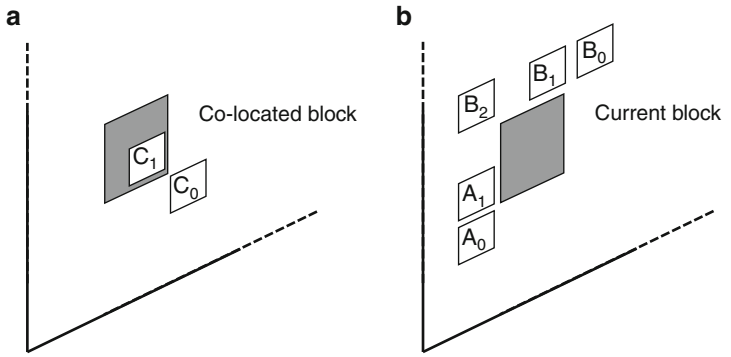


Fig. 5.4 Motion vector predictor and merge candidates. (**a**) Temporal. (**b**) Spatial

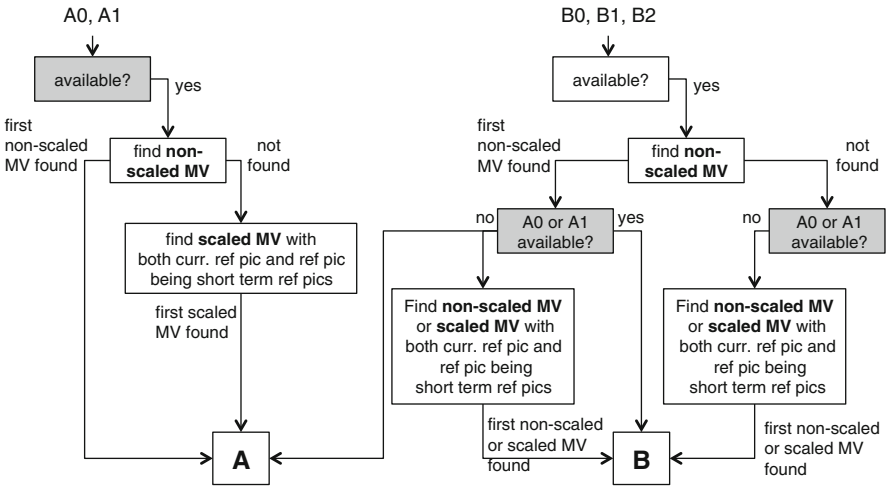


Fig. 5.5 Derivation of spatial AMVP candidates A and B from motion data of neighboring blocks A0, A1, B0, B1 and B2

motion vectors need to be scaled according to the temporal distances between the candidate reference picture and the current reference picture. Equation (5.3) shows how the candidate motion vector $\mathbf{mv}_{\text{cand}}$ is scaled according to a scale factor. ScaleFactor is calculated in Eq. (5.4) based on the temporal distance between the current picture and the reference picture of the candidate block td and the temporal distance between the current picture and the reference picture of the current block tb. The temporal distance is expressed in terms of difference between the picture order count (POC) values which define the display order of the pictures. The scaling operation is basically the same scheme that is used for the temporal direct mode in H.264/AVC. This factoring allows pre-computation of ScaleFactor at slice-level

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00129

since it only depends on the reference picture list structure signaled in the slice header. Note that the MV scaling is only performed when the current reference picture and the candidate reference picture are both short term reference pictures.

$$\mathbf{mv} = \text{sign}(\mathbf{mv}_{\text{cand}} \cdot \text{ScaleFactor}) \cdot ((|\mathbf{mv}_{\text{cand}} \cdot \text{ScaleFactor}| + 2^7) \gg 8) \tag{5.3}$$

$$\text{ScaleFactor} = \text{clip}(-2^{12}, 2^{12} - 1, (\text{tb} \cdot tx + 2^5) \gg 6) \tag{5.4}$$

$$tx = \frac{2^{14} + |\frac{\text{td}}{2}|}{\text{td}} \tag{5.5}$$

For candidate B, the candidates B0 to B2 are checked sequentially in the same way as A0 and A1 are checked in the first pass. The second pass, however, is only performed when blocks A0 and A1 do not contain any motion information, i.e. are not available or coded using intra-picture prediction. Then, candidate A is set equal to the non-scaled candidate B, if found, and candidate B is set equal to a second, non-scaled or scaled variant of candidate B. Since you could also end up in the second pass when there still might be potential non-scaled candidates, the second pass searches for non-scaled as well as for scaled MVs derived from candidates B0 to B2.

Overall, this design allows to process A0 and A1 independently from B0, B1, and B2. The derivation of B should only be aware of the availability of both A0 and A1 in order to search for a scaled or an additional non-scaled MV derived from B0 to B2. This dependency is acceptable given that it significantly reduces the complex motion vector scaling operations for candidate B. Reducing the number of motion vector scalings represents a significant complexity reduction in the motion vector predictor derivation process.

Temporal Candidate

It can be seen from Fig. 5.4b that only motion vectors from spatial neighboring blocks to the left and above the current block are considered as spatial MVP candidates. This can be explained by the fact that the blocks to the right and below the current block are not yet decoded and hence, their motion data is not available. Since the co-located picture is a reference picture which is already decoded, it is possible to also consider motion data from the block at the same position, from blocks to the right of the co-located block or from the blocks below. In HEVC, the block to the bottom right and at the center of the current block have been determined to be the most suitable to provide a good temporal motion vector predictor (TMVP). These candidates are illustrated in Fig. 5.4a where C0 represents the bottom right neighbor and C1 represents the center block. Here again, motion data of C0 is considered first and, if not available, motion data from the co-located candidate block at the center is used to derive the temporal MVP candidate C. The motion data of C0 is also considered as not being available when the associated

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00130

PU belongs to a CTU beyond the current CTU row. This minimizes the memory bandwidth requirements to store the co-located motion data. In contrast to the spatial MVP candidates, where the motion vectors may refer to the same reference picture, motion vector scaling is mandatory for the TMVP. Hence, the same scaling operation from Eq. (5.3) as for the spatial MVPs is used whereas td is defined as the POC difference between the co-located picture and the reference picture of the co-located candidate block.

While the temporal direct mode in H.264/AVC always refers to the first reference picture in the second reference picture list, list 1, and is only allowed in bi-predictive slices, HEVC offers the possibility to indicate for each picture which reference picture is considered as the co-located picture. This is done by signaling in the slice header the co-located reference picture list and reference picture index as well as requiring that these syntax elements in all slices in a picture should specify the same reference picture.

Since the temporal MVP candidate introduces additional dependencies, it might be desirable to disable its usage for error robustness reasons. In H.264/AVC there is the possibility to disable the temporal direct mode for bi-predictive slices in the slice header (`direct_spatial_mv_pred_flag`). HEVC syntax extends this signaling by allowing to disable the TMVP at sequence level or at picture level (`sps/slice_temporal_mvp_enabled_flag`). Although the flag is signaled in the slice header, it is a requirement of bitstream conformance that its value shall be the same for all slices in one picture. Since the signaling of the picture-level flag depends on the SPS flag, signaling it in the PPS would introduce a parsing dependency between SPS and PPS. Another advantage of this slice header signaling is that if you want to change only the value of this flag and no other parameter in the PPS, there is no need to transmit a second PPS.

5.2.1.2 AMVP Motion Data Signaling

In general, motion data signaling in HEVC is similar as in H.264/AVC. An inter-picture prediction syntax element, `inter_pred_idc`, signals whether reference list 0, 1 or both are used. For each MCP obtained from one reference picture list, the corresponding reference picture (Δt) is signaled by an index to the reference picture list, `ref_idx_l0/1`, and the MV (Δx, Δy) is represented by an index to the MVP, `mvp_l0/1_flag`, and its MVD. The MVD syntax is further detailed in Chap. 8. A newly introduced flag in the slice header, `mvd_l1_zero_flag`, indicates whether the MVD for the second reference picture list is equal to zero and therefore not signaled in the bitstream. When the motion vector is fully reconstructed, a final clipping operation assures that the values of each component of the final motion vector will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00131

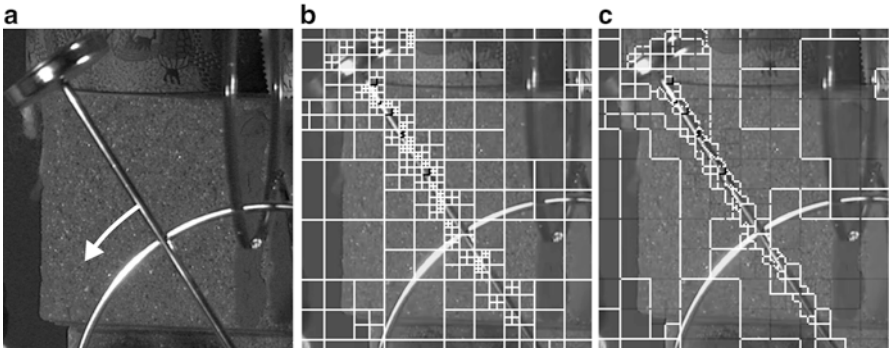


Fig. 5.6 Detail of a video sequence exemplifying the merge concept. (**a**) In the foreground, the scene contains a moving object (a pendulum) with motion indicated by the *arrow*. (**b**) A rate-distortion optimized quadtree partitioning for inter-picture prediction parameters is indicated by the *white borders*. (**c**) Only effective block borders are shown, i.e. borders that separate blocks with different motion parameters. Reproduced with permission from [12], © 2012 IEEE

5.2.2 Inter-picture Prediction Block Merging

In image or video compression it is very reasonable to deploy a block based image partitioning mechanism in order to apply different prediction models to different regions of an image. This is because a single model can in general not be expected to capture the versatile characteristics of a whole image or video. HEVC uses a quadtree structure to describe the partitioning of a region into sub-blocks. In terms of bit rate, this is a very low-cost structure while at the same time, it allows for partitioning into a wide range of differently sized sub-blocks. While this simplicity is an advantage for example for encoder design, it also bears the disadvantage of over-segmenting the image, potentially leading to redundant signaling and ineffective borders. This drawback is effectively addressed by block merging as explained in Sect. 5.2.2.1. A description of the exact algorithm and bitstream syntax follows in Sects. 5.2.2.2–5.2.2.5.

5.2.2.1 Background

An example partitioning using the quadtree structure is shown in Fig. 5.6a for a uni-predictive slice in an HEVC-encoded video. As can be seen, the area around the pendulum is heavily partitioned in order to capture the motion in front of the still background. Figure 5.6c shows the same partitioning, but without ineffective borders, i.e. borders dividing regions of equal motion parameters. It becomes evident that in this particular situation, the quadtree structure is unable to accurately capture the motion without introducing ineffective borders. It is easy to see that this over-segmentation easily occurs whenever moving objects in a scene cause

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00132

abrupt changes in the field of motion parameters, which is common in natural video content. One reason is that the quad-tree structure systematically does not allow for joint description of child blocks that belong to different parent blocks. Also, the fact that a block can only be divided into exactly four child blocks will eventually lead to ineffective borders.

To remedy these inherent drawbacks of the quad-tree structure, HEVC uses block merging which allows us to code motion parameters very cheaply (in terms of bit rate) in these ineffective border situations [9, 12, 23]. The algorithm was inspired by the work of [10], in which the authors show that rate-distortion optimized tree pruning for quadtree-based motion models can be substantially improved by introducing a subsequent leaf merging step. In [22], the authors study the benefits of leaf merging on a broader theoretical basis. Here we leave it at the intuitive example given above and concentrate on the integration of block merging into HEVC. Following from the observation of ineffective borders, block merging introduces a terse syntax allowing for a sub-block to explicitly reuse the exact same motion parameters contained in neighboring blocks. Like AMVP, it compiles a list of candidate motion parameter tuples by picking from neighboring blocks. Then, an index is signaled which identifies the candidate to be used. Block merging also allows for temporal prediction by including into the list a candidate obtained from previously coded pictures. A more detailed description is given in the following.

5.2.2.2 Merge Candidate List Construction

Although they appear similar, there is one main difference between the AMVP and the merge candidate list. The AMVP list only contains motion vectors for one reference list while a merge candidate contains all motion data including the information whether one or two reference picture lists are used as well as a reference index and a motion vector for each list. This significantly reduces motion data signaling overhead. Section 5.2.2.3 describes the signaling in detail as well as discusses how parsing robustness is achieved. Overall, the merge candidate list is constructed based on the following candidates:

- up to four *spatial merge candidates* that are derived from five spatial neighboring blocks
- one *temporal merge candidate* derived from two temporal, co-located blocks
- *additional merge candidates* including combined bi-predictive candidates and zero motion vector candidates

Spatial Candidates

The first candidates in the merge candidate list are the spatial neighbors. Here, the same neighboring blocks as for the spatial AMVP candidates are considered which are described in Sect. 5.2.1.1 and illustrated in Fig. 5.4b. In order to derive a list of motion vector predictors for AMVP, one MVP is derived from A0 and A1 and

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00133

one from B0, B1 and B2, respectively in that order. However, for inter-prediction block merging, up to four candidates are inserted in the merge list by sequentially checking A1, B1, B0, A0 and B2, in that order.

Instead of just checking whether a neighboring block is available and contains motion information, some additional redundancy checks are performed before taking all the motion data of the neighboring block as a merge candidate. These redundancy checks can be divided into two categories for two different purposes:

- avoid having candidates with redundant motion data in the list
- prevent merging two partitions that could be expressed by other means which would create redundant syntax

When N is the number of spatial merge candidates, a complete redundancy check would consist of $\frac{N \cdot (N-1)}{2}$ motion data comparisons. In case of the five potential spatial merge candidates, ten motion data comparisons would be needed to assure that all candidates in the merge list have different motion data. During the development of HEVC, the checks for redundant motion data have been reduced to a subset in a way that the coding efficiency is kept while the comparison logic is significantly reduced [1]. In the final design, no more than two comparisons are performed per candidate resulting in five overall comparisons. Given the order of {A1, B1, B0, A0, B2}, B0 only checks B1, A0 only A1 and B2 only A1 and B1.

For an explanation of the partitioning redundancy check consider the following example. The bottom PU of a 2N×N partitioning is merged with the top one by choosing candidate B1. This would result in one CU with two PUs having the same motion data which could be equally signaled as a 2N×2N CU. Overall, this check applies for all second PUs of the rectangular and asymmetric partitions 2N×N, 2N×nU, 2N×nD, N×2N, nR×2N and nL×2N. Please note that for the spatial merge candidates, only the redundancy checks are performed and the motion data is copied from the candidate blocks as it is. Hence, no motion vector scaling is needed here.

Temporal Candidate

The derivation of the motion vectors for the temporal merge candidate is the same as for the TMVP described in Sect. 5.2.1.1. Since a merge candidate comprises all motion data and the TMVP is only one motion vector, the derivation of the whole motion data only depends on the slice type. For bi-predictive slices, a TMVP is derived for each reference picture list. Depending on the availability of the TMVP for each list, the prediction type is set to bi-prediction or to the list for which the TMVP is available. All associated reference picture indices are set equal to zero. Consequently for uni-predictive slices, only the TMVP for list 0 is derived together with the reference picture index equal to zero.

When at least one TMVP is available and the temporal merge candidate is added to the list, no redundancy check is performed. This makes the merge list construction independent of the co-located picture which improves error resilience. Consider the case where the temporal merge candidate would be redundant and therefore not

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00134

Table 5.1 Order in which motion data combinations of different merge candidates, that have been already inserted in the merge list, are tested to create combined bi-predictive merge candidates

Combination Order	0	1	2	3	4	5	6	7	8	9	10	11
$\Delta x_0, \Delta y_0, \Delta t_0$ from Cand.	0	1	0	2	1	2	0	3	1	3	2	3
$\Delta x_1, \Delta y_1, \Delta t_1$ from Cand.	1	0	2	0	2	1	3	0	3	1	3	2

included in the merge candidate list. In the event of a lost co-located picture, the decoder could not derive the temporal candidates and hence not check whether it would be redundant. The indexing of all subsequent candidates would be affected by this.

Additional Candidates

For parsing robustness reasons, which will be explained in Sect. 5.2.2.3, the length of the merge candidate list is fixed. After the spatial and the temporal merge candidates have been added, it can happen that the list has not yet the fixed length. In order to compensate for the coding efficiency loss that comes along with the non-length adaptive list index signaling, additional candidates are generated [25]. Depending on the slice type, up to two kind of candidates are used to fully populate the list:

- Combined bi-predictive candidates
- Zero motion vector candidates

In bi-predictive slices, additional candidates can be generated based on the existing ones by combining reference picture list 0 motion data of one candidate with and the list 1 motion data of another one. This is done by copying $\Delta x_0, \Delta y_0, \Delta t_0$ from one candidate, e.g. the first one, and $\Delta x_1, \Delta y_1, \Delta t_1$ from another, e.g. the second one. The different combinations are predefined and given in Table 5.1.

When the list is still not full after adding the combined bi-predictive candidates, or for uni-predictive slices, zero motion vector candidates are calculated to complete the list. All zero motion vector candidates have one zero displacement motion vector for uni-predictive slices and two for bi-predictive slices. The reference indices are set equal to zero and are incremented by one for each additional candidate until the maximum number of reference indices is reached. If that is the case and there are still additional candidates missing, a reference index equal to zero is used to create these. For all the additional candidates, no redundancy checks are performed as it turned out that omitting these checks will not introduce a coding efficiency loss [21].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00135

5.2.2.3 Merge Motion Data Signaling and Skip Mode

The motion data signaling scheme using the merge mode is quite simple. For each PU coded in inter-picture prediction mode, a so called `merge_flag` indicates that block merging is used to derive the motion data. The `merge_idx` further determines the candidate in the merge list that provides all the motion data needed for the MCP. Therefore, instead of all the syntax elements needed for AMVP based motion data coding described in Sect. 5.2.1.2, only a flag and a list index are transmitted. This difference can be seen when comparing the input to the AMVP and the merge motion data coding block in Fig. 5.2.

Besides this PU-level signaling, the number of candidates in the merge list is signaled in the slice header. Since the default value is five, it is represented as a difference to five (`five_minus_max_num_merge_cand`). That way, the five is signaled with a short codeword for the 0 whereas using only one candidate, is signaled with a longer codeword for the 4. Regarding the impact on the merge candidate list construction process, the overall process remains the same although it terminates after the list contains the maximum number of merge candidates. In the initial design, the maximum value for the merge index coding was given by the number of available spatial and temporal candidates in the list. When e.g. only two candidates are available, the index can be efficiently coded as a flag. But, in order to parse the merge index, the whole merge candidate list has to be constructed to know the actual number of candidates. Assuming unavailable neighboring blocks due to transmission errors, it would not be possible to parse the merge index anymore. Fixing the number of merge candidates improves the parsing robustness by decoupling the parsing and the merge candidate list construction while sacrificing coding efficiency. Populating the list with the additional merge candidates presented in Sect. 5.2.2.2 compensates again for that loss while keeping the parsing robustness.

A crucial application of the block merging concept in HEVC is its combination with a skip mode. In previous video coding standards, the skip mode was used to indicate for a block that the motion data is inferred instead of explicitly signaled and that the prediction residual is zero, i.e. no transform coefficients are transmitted. This mode is well suited to code static image regions where the prediction error tends to be very small. In HEVC, at the beginning of each CU in an inter-picture prediction slice, a `skip_flag` is signaled that implies the following:

- the CU only contains one PU (2N×2N partition type)
- the merge mode is used to derive the motion data (`merge_flag` equal to 1)
- no residual data is present in the bitstream

5.2.2.4 Coding Efficiency of HEVC Merge and Skip Mode

In this section, the coding efficiency of the HEVC merge and skip modes is analyzed. This is done experimentally by disabling the merge mode as well as the skip mode, i.e. removing `merge_flag`, `merge_index` and `skip_flag`

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00136

Table 5.2 Average bit rate savings of HEVC merge and skip mode using HM8.0

		BD-rate [%]		
Class	Sequence	RA-Main	LB-Main	LD-Main
A	Traffic	−8.8	n/a	n/a
(2560 × 1600)	PeopleOnStreet	−6.5	n/a	n/a
	Nebuta	−1.5	n/a	n/a
	SteamLocomotive	−11.1	n/a	n/a
B	Kimono	−9.6	−7.8	−5.4
(1920 × 1080)	ParkScene	−7.3	−7.0	−5.6
	Cactus	−11.7	−8.8	−6.9
	Basketball Drive	−9.1	−8.1	−6.1
	BQTerrace	−10.2	−10.3	−5.9
C	Basketball Drill	−7.5	−7.6	−6.1
(832 × 480)	BQMall	−9.2	−7.5	−5.7
	PartyScene	−4.8	−3.4	−2.5
	RaceHorses	−3.9	−4.3	−3.3
D	Basketball Pass	−6.1	−5.1	−4.0
(416 × 240)	BQSquare	−6.9	−3.3	−2.5
	BlowingBubbles	−6.1	−3.7	−3.1
	RaceHorses	−4.4	−3.9	−3.6
E	FourPeople	n/a	−11.4	−8.5
(1280 × 720)	Johnny	n/a	−20.0	−16.4
	KristenAndSara	n/a	−15.0	−11.2
Average		**−7.3**	**−8.0**	**−6.0**

syntax. The software used for this experiment is version 8.0 of the HEVC test model reference software (HM) [13] with the random access main, low delay B and P main coding configurations as described in [5].

The coding efficiency gains in terms of Bjøntegaard Delta (BD) rate [2], when enabling the merge and skip modes, are reported in [9] and [12] and also summarized in Table 5.2. Average bit rate savings between 6 % and 8 % are observed. It can be seen that the gains for the random access main (RA-Main) and low delay B main (LB-Main) configurations using bi-predictive B pictures are higher than for the low delay P main (LP-Main) configuration, which is restricted to use uni-prediction. Since the merge mode only uses a flag and an index to signal all motion data, it is more efficient the more motion data signaling is omitted that way, e.g. two sets of motion data in bi-prediction. Another observation is that merge and skip modes are saving up to 20 % for the class E sequences. These sequences represent videoconferencing content where the static background can efficiently be coded using skip and merge modes. More detailed results and a comparison with an AMVP-based direct mode can be found in [12].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00137

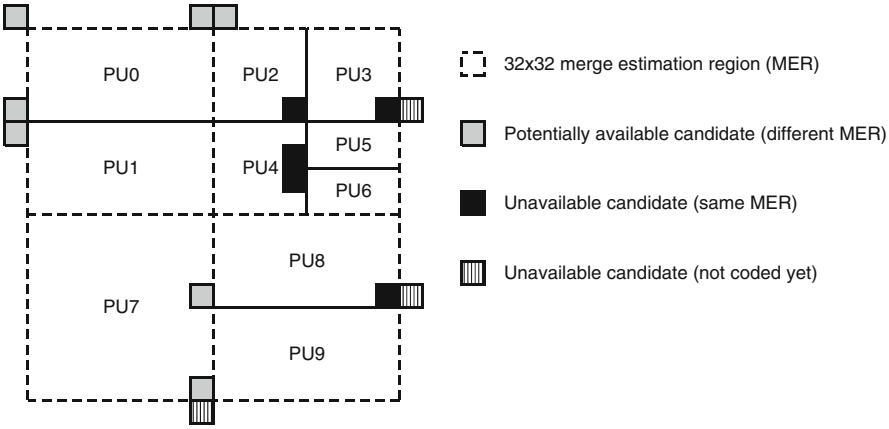


Fig. 5.7 Example of a CTU with a 64×64 luma CTB, when motion estimation of for PUs inside a 32×32 motion estimation region is carried out independently, enabling the possibility to do it in parallel

5.2.2.5 Merge Estimation Regions for Parallel Merge Mode Estimation

The way the merge candidate list is constructed introduces dependencies between neighboring blocks. Especially in embedded encoder implementations, the motion estimation stage of neighboring blocks is typically performed in parallel or at least pipelined to increase the throughput. For AMVP, this is not a big issue since the MVP is only used to differentially code the MV found by the motion search. The motion estimation stage for the merge mode, however, would typically just consist of the candidate list construction and the decision which candidate to choose, based on a cost function. Due to the aforementioned dependency between neighboring blocks, merge candidate lists of neighboring blocks cannot be generated in parallel and represent a bottleneck for parallel encoder designs. Therefore, a parallel merge estimation level was introduced in HEVC that indicates the region in which merge candidate lists can be independently derived by checking whether a candidate block is located in that merge estimation region (MER). A candidate block that is in the same MER is not included in the merge candidate list. Hence, its motion data does not need to be available at the time of the list construction. When this level is e.g. 32, all prediction units in a 32×32 area can construct the merge candidate list in parallel since all merge candidates that are in the same 32×32 MER, are not inserted in the list. Figure 5.7 illustrates that example showing a CTU partitioning with seven CUs and ten PUs. All potential merge candidates for the first PU0 are available because they are outside the first 32×32 MER. For the second MER, merge candidate lists of PUs 2–6 cannot include motion data from these PUs when the merge estimation inside that MER should be independent. Therefore, when looking at a PU5 for example, no merge candidates are available and hence not inserted in

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00138

Table 5.3 Average bit rate losses for different merge estimation regions in terms of BD-rate using HM5.0 reference software

	Merge estimation region			
	8 × 8	16 × 16	32 × 32	64 × 64
RA-HE	0.1 %	0.6 %	1.6 %	2.7 %
LB-HE	0.2 %	0.7 %	2.0 %	3.4 %

the merge candidate list. In that case, the merge list of PU5 consists only of the temporal candidate (if available) and zero MV candidates.

In order to enable an encoder to trade-off parallelism and coding efficiency, the parallel merge estimation level is adaptive and signaled as `log2_parallel_merge_level_minus2` in the picture parameter set. The following MER sizes are allowed: 4×4 (no parallel merge estimation possible), 8×8, 16×16, 32×32 and 64 × 64. A higher degree of parallelization, enabled by a larger MER, excludes more potential candidates from the merge candidate list. That, on the other hand, decreases the coding efficiency. The coding efficiency losses in terms of BD-rate [2] for different MER sizes are reported in [28] and summarized in Table 5.3. Results are generated using HM5.0 [13] with random access high efficiency (RA-HE) and low delay B high efficiency (LB-HE) coding configurations as described in [3].

When the merge estimation region is larger than a 4×4 block, another modification of the merge list construction to increase the throughput kicks in. For a CU with an 8×8 luma CB, only a single merge candidate list is used for all PUs inside that CU.

5.2.3 Motion Data Storage Reduction

The usage of the TMVP, in AMVP as well as in the merge mode, requires the storage of the motion data (including motion vectors, reference indices and coding modes) in co-located reference pictures. Considering the granularity of motion representation, the memory size needed for storing motion data could be significant. HEVC employs *motion data storage reduction* (MDSR) to reduce the size of the motion data buffer and the associated memory access bandwidth by sub-sampling motion data in the reference pictures [20]. While H.264/AVC is storing these information on a 4×4 block basis, HEVC uses a 16×16 block where, in case of sub-sampling a 4×4 grid, the information of the top-left 4×4 block is stored. Due to this sub-sampling, MDSR impacts on the quality of the temporal prediction. Furthermore, there is a tight correlation between the position of the MV used in the co-located picture, and the position of the MV stored by MDSR.

During the standardization process of HEVC, the impact of the sub-sampling scheme as well as the interaction with the TMVP was investigated in a core experiment on MDSR [15]. It turned out that storing the motion data of the top left block inside the 16×16 area together with the bottom right and center

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00139

TMVP candidates provide the best tradeoff between coding efficiency and memory bandwidth reduction. Furthermore, the general impact of sub-sampling the motion information was measured. While the 8×8 subsampling show no difference in coding efficiency compared to 4×4, the current 16×16 scheme results in a coding efficiency loss of 0.1 % BD-rate which is negligible and can be considered as being in the noise margin.

5.3 Fractional Sample Interpolation

Interpolation tasks arise naturally in the context of video coding because the true displacements of objects from one picture to another are independent of the sampling grid of cameras. Therefore, in MCP, fractional-sample accuracy is used to more accurately capture continuous motion. Samples available at integer positions are filtered to estimate values at fractional positions. This spatial domain operation can be seen in the frequency domain as introducing phase delays to individual frequency components. An ideal interpolation filter for band-limited signals induces a constant phase delay to all frequencies and does not alter their magnitudes. The efficiency of MCP is limited by many factors—the spectral content of original and already reconstructed pictures, camera noise level, motion blur, quantization noise in reconstructed pictures, etc.

Similar to H.264/AVC, HEVC supports motion vectors with quarter-pixel accuracy for the luma component and one-eighth pixel accuracy for chroma components. If the motion vector has a half or quarter-pixel accuracy, samples at fractional positions need to be interpolated using the samples at integer-sample positions. The interpolation process in HEVC introduces several improvements over H.264/AVC that contributes to the significant coding efficiency increase of HEVC. In this section, these differences are first explained and then the complexity and coding efficiency characteristics of the HEVC interpolation process are presented.

5.3.1 Overview

In order to improve the filter response in the high frequency range, luma and chroma interpolation filters have been re-designed and the tap-lengths were increased. The luma interpolation process in HEVC uses a symmetric 8-tap filter for half-sample positions and an asymmetric 7-tap filter for quarter-sample positions. For chroma samples, a 4-tap filter was introduced.

The intermediate values used in interpolation process are kept at a higher accuracy in HEVC to improve coding efficiency. This is done as follows (please refer to Fig. 5.9 for notation throughout the text, where integer-sample values are shown with dark squares and the fractional-sample values are shown with white squares):

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00140

- H.264/AVC obtains the quarter-sample values by first obtaining the values of nearest half-pixel samples and averaging those with the nearest integer samples, according the position of the quarter-pixel [27]. However, HEVC obtains the quarter-pixel samples without using such cascaded steps but by instead directly applying a 7 or 8-tap filter on the integer pixels.
- In H.264/AVC, a bi-predictively coded block is calculated by averaging two uni-predicted blocks. If interpolation is performed to obtain the samples of the uni-prediction blocks, those samples are shifted and clipped to input bit-depth after interpolation, prior to averaging. On the other hand, HEVC keeps the samples of each one of the uni-prediction blocks at a higher accuracy and only performs rounding to input bit-depth at the final stage, improving the coding efficiency by reducing the rounding error.

The details of these features are presented in the following sections.

5.3.1.1 Redesigned Filters

An important parameter for interpolation filters is the number of filter taps as it has a direct influence on both coding efficiency and implementation complexity. In terms of implementation, it not only has an impact on the arithmetic operations but also on the memory bandwidth required to access the reference samples. Although the 6-tap filter for estimating half-pixel positions in H.264/AVC produces a constant phase delay of 0.5 for all frequency components due to symmetry, the passband (range of frequencies where the magnitudes are relatively unaltered) is not large. Increasing the number of taps can yield filters that produce desired response for a larger range of frequencies which can help to predict the corresponding frequencies in the samples to be coded. Considering modern computing capabilities, the performance of many MCP filters were evaluated in the context of HEVC and a coding efficiency/complexity trade-off was targeted during the standardization.

Consider the design of a half-pixel interpolation filter with $2N$ taps denoted as $\mathbf{h} = [h_0, h_1, \cdots, h_{2N-1}]^T$. Due to the desired half-pixel symmetry only N coefficients can be different, which can be denoted in the form $\mathbf{h} = [h_0, h_1, \cdots, h_{N-1}, \cdots, h_1, h_0]^T$. Now consider the interpolation of a DC signal (with all samples equal). It is desired that the interpolated value be the same as the input. Hence, we require $2 \cdot \sum_{n=0}^{N-1} h_n = 1$, also known as the normalization constraint. This further reduces the number of degrees of freedom from N to $N-1$. In the case of a quarter-pixel interpolation filter however, only the normalization condition $\sum_{n=0}^{2N-1} h_n = 1$ is imposed as symmetry is not necessary, which gives $2N-1$ degrees of freedom. The aim of interpolation filter design is to determine these degrees of freedom so as to remain close to the desired frequency response.

Here a brief overview of the design of HEVC interpolation filters is provided. For a detailed explanation the reader is referred to [17]. The basic idea is to forward transform the known integer samples to the DCT domain and inverse transform

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00141

Table 5.4 Filter coefficients for luma interpolation in MCP

Phase	Luma filter coefficients
1/4	$[-1, 4, -10, 58, 17, -5, 1]/64$
1/2	$[-1, 4, -11, 40, 40, -11, 4, -1]/64$

Table 5.5 Filter coefficients for chroma interpolation in MCP

Phase	Chroma filter coefficients
1/8	$[-2, 58, 10, -2]/64$
1/4	$[-4, 54, 16, -2]/64$
3/8	$[-6, 46, 28, -4]/64$
1/2	$[-4, 36, 36, -4]/64$

the DCT coefficients to the spatial domain using DCT basis sampled at desired fractional positions instead of integer positions. Fortunately, these operations can be combined into a single FIR filtering step. Let the available samples at integer positions be denoted as a column vector $\mathbf{s}$ and the forward transform as a matrix $\mathbf{B}$. The DCT coefficients $\mathbf{c}$ can be computed as $\mathbf{c} = \mathbf{B} \cdot \mathbf{s}$. Since DCT basis is composed of cosine functions (which are continuous in nature), they can be sampled at fractional positions. Let the DCT basis sampled at desired fractional positions be denoted by a row vector $\mathbf{r}$. This is used to transform back the DCT coefficients to the spatial domain, which results in the interpolated value $\hat{s} = \mathbf{r} \cdot \mathbf{B} \cdot \mathbf{s}$. These stages can be combined into a single filter $\mathbf{f} = \mathbf{r} \cdot \mathbf{B}$. In addition to the above steps, the reference samples are smoothed in the actual HEVC design to combat noise in the reference samples. Therefore the final interpolation filter can be written in the form $\mathbf{f} = \mathbf{r} \cdot \mathbf{B} \cdot \mathbf{W}$, where $\mathbf{W}$ is a diagonal matrix with weights for smoothing. The resulting filter coefficients are rounded to 6-bit precision (for a simple fixed-point implementation) and an integer optimization is carried out under the normalization constraint to ensure that the filter coefficients provide close to desired frequency response even after rounding. For the chroma interpolation filters, a slightly different smoothing of reference samples is performed during the filter design. The filter coefficients' bit-depth of 6 also makes it possible to realize the entire MCP process for 8-bit videos using 16-bit intermediate buffers. The filter coefficients resulting from the design described above for luma and chroma MCP are given in Tables 5.4 and 5.5, respectively. The magnitude responses of half-pel luma interpolation filters of H.264/AVC and HEVC are depicted in Fig. 5.8. It can be seen that the half-pel interpolation filter of HEVC comes closer to the desired response than the H.264/AVC filter.

5.3.1.2 High Precision Filtering Operations

In H.264/AVC, some of the intermediate values used within interpolation are shifted to lower accuracy, which introduces rounding error and reduces coding efficiency. This loss of accuracy is due to several reasons. Firstly, the half-pixel

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00142

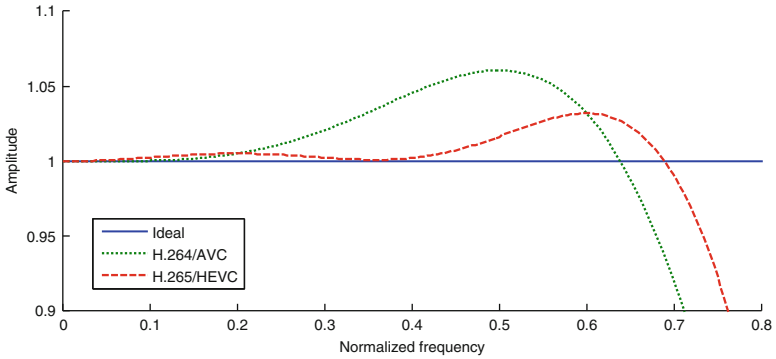


Fig. 5.8 Comparison of magnitude response of half-pel interpolation filters of H.264/AVC and HEVC relative to an ideal filter

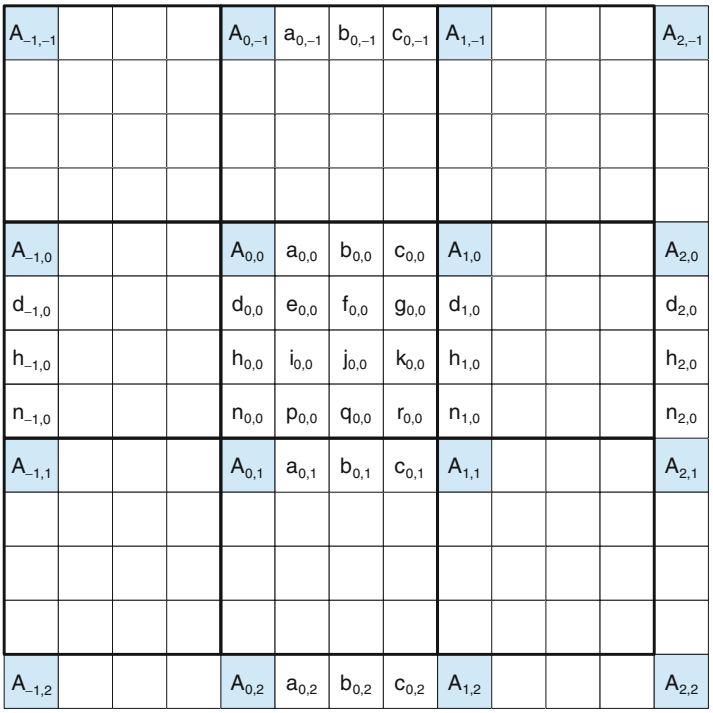


Fig. 5.9 Fractional positions used in luma motion compensation with 1/4 pixel accuracy

samples obtained by 6-tap FIR filter are first rounded to input bit-depth, prior to using those for obtaining the quarter-pixel samples. To illustrate this effect, let's consider interpolating the quarter-pixel sample $a_{0,0}$ using the H.264/AVC interpolation process. In order to obtain $a_{0,0}$, the half-pixel sample $b_{0,0}$ needs to

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00143

be obtained first by applying a 6-tap horizontal FIR filter and rounding the result to input bit-depth, as shown in Eq. (5.6). The quarter-pixel sample $a_{0,0}$ is then obtained by averaging the half-pixel sample, $b_{0,0}$ with the integer sample $A_{0,0}$ as shown in Eq. (5.7). Because $b_{0,0}$ is first rounded back to input bit-depth, a rounding error of $33/128$ is introduced to obtain $a_{0,0}$ [16].

$$b_{0,0} = (A_{-2,0} - 5 \cdot A_{-1,0} + 20 \cdot A_{0,0} + 20 \cdot A_{1,0} - 5 \cdot A_{2,0} + A_{3,0} + 16) >> 5 \tag{5.6}$$

$$a_{0,0} = (A_{0,0} + b_{0,0} + 1) >> 1 \tag{5.7}$$

Instead of using a two-stage cascaded filtering process, HEVC interpolation filter computes the quarter-pixels directly using a 7-tap filter using the coefficients shown in Sect. 5.3.1.1, which significantly reduces the rounding error to $1/128$.

The second reason for reduction of accuracy in H.264/AVC motion compensation process is due to averaging in bi-prediction. In H.264/AVC, the prediction signal of the bi-predictively coded motion blocks (denoted by S) is obtained by averaging prediction signals from two prediction lists (denoted by S_1 and S_2) as shown in Eq. (5.8).

$$S = (S_1 + S_2 + 1) >> 1 \tag{5.8}$$

The averaging operation shown in Eq. (5.8) is done at the precision of input bit-depth (i.e. S_1 and S_2 are 8-bit for an 8-bit video). If the motion vectors have fractional pixel accuracy, then S_1 and S_2 are obtained using interpolation and the intermediate values are rounded to input bit-depth. In HEVC, instead of averaging each prediction signal at the precision of the bit-depth, they are averaged at a higher precision if fractional motion vectors are used for the corresponding block [18]. This means that, if the motion vectors to obtain S_1 or S_2 have sub-pixel accuracy, the interpolation process does not round the intermediate values to input-bit depth prior to averaging, but keeps it at a higher precision. It should be noted that for the cases where one of the prediction signal is obtained without interpolation (i.e. the corresponding motion vector has an integer pixel accuracy) the bit-depth of the corresponding prediction signal is first increased accordingly before bi-prediction averaging so that both prediction signals are averaged at the same bit-depth.

This process is illustrated in Fig. 5.10 (a) for the case of H.264/AVC where bi-prediction averages two prediction signals at input bit-depth and (b) for HEVC where the averaging is performed at a higher bit-depth and intermediate rounding step is not used.

5.3.1.3 Other Important Features

To make sure the intermediate values do not overflow the 16-bit registers, after horizontal interpolation the intermediate values are shifted to the right by bit depth minus 2. This means that when the bit depth of the video is more than 8 bits, the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00144

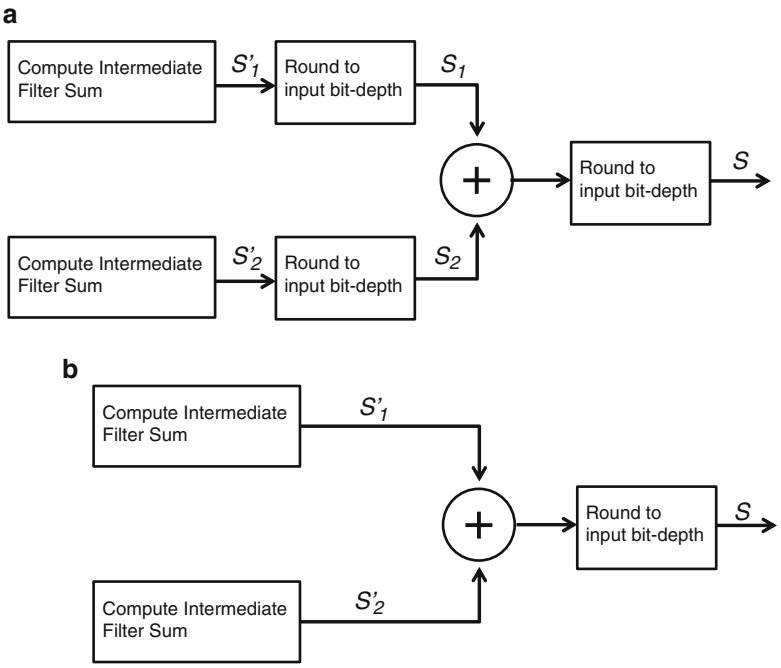


Fig. 5.10 Bi-prediction process in (**a**) H.264/AVC and in (**b**) HEVC. Reproduced with permission from [17], © 2013 IEEE

order in which horizontal filtering and vertical filtering is done needs to be specified (horizontal first in HEVC). This specific order was selected mainly to simplify implementation on specific architectures.

It should also be noted that in HEVC the only clipping operation is at the very end of the motion compensation process, with no clipping in intermediate stages. As there is also no rounding in intermediate stages, HEVC interpolation filter allows certain implementation optimizations. Consider the case where bi-prediction is used and motion vectors of each prediction direction points to the same fractional position. In these cases, final prediction could be obtained by first adding two reference signals and performing interpolation and rounding once, instead of interpolating each reference block, thus saving one interpolation process.

5.3.2 Complexity and Coding Efficiency Characteristics

In this section, the complexity of the interpolation filter design in HEVC is analyzed and compared with that of H.264/AVC. In addition, the coding efficiency improvements brought with the improved design are also presented.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00145

5.3.2.1 Complexity of HEVC Interpolation Filter

When evaluating the complexity of a video coding algorithm, several aspects, such as memory bandwidth, number of operations and storage buffer size need to be carefully considered.

In terms of memory bandwidth, utilizing longer tap filters in HEVC (7–8 tap filter for luma sub-pixels and 4-tap filter for chroma sub-pixels) compared to shorter filters in H.264/AVC (6-tap filter for luma sub-pixels and bilinear filter for chroma) increases the amount of data that needs to be fetched from the reference memory. The worst case happens when a small motion block is bi-predicted and its corresponding motion vector points to a sub-pixel position where two-dimensional filtering needs to be performed (such as position $f_{0,0}$). In order to reduce the worst case memory bandwidth, HEVC introduces several restrictions. Firstly, the smallest prediction block size is fixed to be 4×8 or 8×4, instead of 4×4. In addition, these smallest block sizes of size 4×8 and 8×4 can only be predicted with uni-prediction. With these restrictions in place, the worst-case memory bandwidth of HEVC interpolation filter is around 51 % higher than that of H.264/AVC. The increase in memory bandwidth is not very high for larger block sizes. For example, for a 32×32 motion block, HEVC requires around 13 % increased memory bandwidth over H.264/AVC [17].

Similarly, the longer tap-length filters increase the number of arithmetic operations required to obtain the interpolated sample. If the complexity is measured by the number of multiply-and-add operations (MACs), interpolation filter in HEVC represents roughly a 20 % increase over H.264/AVC filter for 8-bit video.

The high-precision bi-directional averaging described in Sect. 5.3.1.2 increases the size of intermediate storage buffers for storing the temporary uni-prediction signals as each one of the prediction signals need to be stored at a higher bit-depth compared to H.264/AVC before the bi-directional averaging takes place.

HEVC uses 7-tap FIR filter for interpolating samples at quarter-pixel locations, which has an impact on motion estimation of a video encoder. An H.264/AVC encoder could store only the integer and half-pel samples in the memory and generate the quarter-pixels on-the-fly during motion estimation. This would be significantly more costly in HEVC because of the complexity of generating each quarter-pixel sample on-the-fly with a 7-tap FIR filter. Instead, an HEVC encoder could store the quarter-pixel samples in addition to integer and half-pixel samples and use those in motion estimation. Alternatively, an HEVC encoder could estimate the values of quarter-pixel samples during motion estimation by low complexity non-normative means.

5.3.2.2 Coding Efficiency of HEVC Interpolation Filter

In this section, the coding efficiency of interpolation filter design in HEVC is analyzed. For this purpose, the H.264/AVC interpolation filter is first implemented in version 6.0 of the HEVC test model and then run with the test conditions advised by JCT-VC [4]. Same test model is also run with the HEVC interpolation filter

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00146

Table 5.6 Average bit rate savings of HEVC interpolation filter for the luma component using HM6.0

		BD-rate [%]		
Class	Sequence	RA-Main	LB-Main	LD-Main
A	Traffic	−2.4	n/a	n/a
(2560 × 1600)	PeopleOnStreet	0.1	n/a	n/a
	Nebuta	0.4	n/a	n/a
	SteamLocomotive	−1.0	n/a	n/a
B	Kimono	−1.8	−2.6	0.6
(1920 × 1080)	ParkScene	−2.7	−4.5	−2.2
	Cactus	−1.3	−2.9	−0.8
	Basketball Drive	−2.0	−3.1	−0.2
	BQTerrace	−5.0	−7.1	0.5
C	Basketball Drill	−2.5	−3.6	−2.1
(832 × 480)	BQMall	−4.3	−5.5	−2.7
	PartyScene	−10.7	−11.9	−10.5
	RaceHorses	−1.2	−2.2	−0.1
D	Basketball Pass	−1.8	−2.7	−1.0
(416 × 240)	BQSquare	−21.6	−21.7	−18.0
	BlowingBubbles	−7.5	−9.1	−7.6
	RaceHorses	−1.8	−3.4	−2.3
E	FourPeople	n/a	−2.0	0.6
(1280 × 720)	Johnny	n/a	−6.8	−2.2
	KristenAndSara	n/a	−3.9	−0.5
Average		−4.0	−4.9	−2.6

and the results are compared. This experiment is conducted to see how much gain collectively all the improvements the HEVC interpolation filter brings. The test conditions can be summarized as:

- Four quantization values used: 22, 27, 32 and 37
- A total number of 24 different sequences are coded. These sequences are divided into different classes that represent different use-cases and video characteristics.
- Tests are conducted for three different prediction structures: Random Access, Low Delay with B pictures and Low Delay with P pictures.
- The coding efficiency is measured by using the Bjøntegaard-Delta bit rate measure [2].

The detailed results are shown in Table 5.6 for the luma component, where it is shown that on average, the interpolation filter of HEVC brings 4.0 % coding efficiency gain. The results for the chroma component is also summarized in Table 5.7 where the average coding efficiency gains reach 11.27 %. For some sequences, especially for those that contain more high frequency content, gains

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00147

Table 5.7 Average bit rate savings of HEVC interpolation filter for the chroma component

	BD-rate [%]					
	RA-Main		LB-Main		LD-Main	
Class	Cb	Cr	Cb	Cr	Cb	Cr
A (2560 × 1600)	−10.8	−11.3	n/a	n/a	n/a	n/a
B (1920 × 1080)	−10.8	−12.2	−14.7	−16.8	−5.8	−6.5
C (832 × 480)	−10.3	−10.8	−13.0	−13.5	−8.9	−9.6
D (416 × 240)	−15.3	−17.1	−20.4	−21.7	−16.6	−18.1
E (1280 × 720)	−2.4	−2.5	−5.4	−4.7	−2.5	−2.8
Average	**−11.7**	**−12.8**	**−13.3**	**−14.3**	**−7.4**	**−8.1**

become very large and become more than 20 %. Further experiments show that close to half of this gain is due to high precision filter operations as described in Sect. 5.3.1.2 and the rest of the gain is due to improved filter coefficients with longer tap-lengths.

5.4 Weighted Sample Prediction

Similar to H.264/AVC, HEVC includes a weighted prediction (WP) tool that is particularly useful for coding sequences with fades. In WP, a multiplicative weighting factor and an additive offset are applied to the motion compensated prediction. In principle, WP replaces the inter prediction signal P by a linearly weighted prediction signal $\hat{P} = w \times P + o$, where w is an Illumination Compensation weight and o is an offset. Care is taken to handle uni-prediction and bi-prediction weights appropriately using the flags `weighted_pred_flag` and `weighted_bipred_flag` transmitted in the Picture Parameter Set (PPS). Consequently, WP has a very small overhead in PPS and slice headers contain only non-default WP scaling values. WP is an optional PPS parameter and it may be switched on/off when necessary. The inputs to the WP process are: the width and the height of the luma prediction block, prediction samples to be weighted, the prediction list utilization flags, the reference indices for each list, and the color component index. Weighting factors w_0 and w_1, and offsets o_0 and o_1 are determined using the data transmitted in the bitstream. The subscripts indicate the reference picture list to which the weight and the offset are applied. The output of this process is the array of prediction sample values. The WP process, for the case when only the list L0 is used, can be written in a simplified form as:

$$\hat{P}[x][y] = \text{Clip3}(0, \text{max\_val}, P_{L0}[x][y] * w_0 + o_0) \tag{5.9}$$

where x and y denote spatial coordinates within the prediction block and max_val represents the maximum value in the considered bit depth. Additionally, a log

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00148

weight denominator (LWD) rounding factor may be used before adding the offset. For the case of bi-prediction, the value to be clipped is calculated as follows (similar rounding offset is used for uni-prediction case as well):

$$(P_{L0}[x][y] * w_0 + P_{L1}[x][y] * w_1 + (o_0 + o_1 + 1) \ll \text{LWD}) \gg (\text{LWD} + 1) \quad (5.10)$$

In H.264/AVC, weight and offset parameters are either derived by relative distances between the current picture and the reference distances (implicit mode) or weight and offset parameters are explicitly signaled (explicit mode) [6]. Unlike H.264/AVC, HEVC only includes explicit mode as the coding efficiency provided by deriving the weighted prediction parameters with implicit mode was considered negligible.

It should be noted that the weighted prediction process defined in HEVC version 1 was found to be not optimal for higher bit-depths as the offset parameter is calculated at low precision. The next version of the standard will likely modify the derivation of the offset parameter for higher bit-depths [24].

The determination of appropriate WP parameters in an encoder is outside the scope of the HEVC standard. Several algorithms for estimating WP parameters have been proposed in literature. Optimal solutions are obtained when the Illumination Compensation weights, motion estimation and Rate Distortion Optimization (RDO) are considered jointly. However, practical systems usually employ simplified techniques, such as determining approximate weights by considering picture-to-picture mean variation.

5.5 Summary and Conclusions

The inter-picture prediction part of the HEVC video coding standard is not introducing a revolutionary whole new design. Moreover, it can be seen as a steady improvement and generalization of all parts known from previous video coding standards, e.g. H.264/AVC. The motion vector prediction was enhanced with advanced motion vector prediction based on motion vector competition. An inter-prediction block merging technique significantly simplified the block-wise motion data signaling by inferring all motion data from already decoded blocks. When it comes to interpolation of fractional reference picture samples, high precision interpolation filter kernels with extended support, i.e. 7/8-tap filter kernels for luma and 4-tap filter kernels for chroma, improve the filtering especially in the high frequency range. Finally, the weighted prediction signaling was simplified by either applying explicitly signaled weights for each motion compensated prediction or just averaging two motion compensated predictions.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00149

References

1. Bici O, Lainema J, Ugur K (2012) CE9: Results of SP experiments on simplification of merge process, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0252, San Jose, Feb. 2012
2. Bjøntegaard G (2001) Calculation of average PSNR differences between RD curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001
3. Bossen F (2011) HM 5 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1200, Geneva, Nov. 2011
4. Bossen F (2012a) HM 6 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H1100, San Jose, Feb. 2012
5. Bossen F (2012b) HM 8 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J1100, Stockholm, July 2012
6. Boyce J (2004) Weighted prediction in the H.264/MPEG AVC video coding standard. In: Proceedings of the 2004 international symposium on circuits and systems, ISCAS '04, vol 3, pp III–789–92, 2004
7. Bross B, Jung J, Chien WJ, Kim IK, Zhou M (2011) CE9: Summary report of core experiment on MV coding and skip/merge operations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G039, Geneva, Nov. 2011
8. Bross B, Jung J, Huang YW, Tan YH, Kim IK, Sugio T, Zhou M, Tan TK, Francois E, Kazui K, Chien WJ, Sekiguchi S, Park S, Wan W (2011) BoG report of CE9: MV Coding and Skip/Merge operations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E481, Geneva, Mar. 2011
9. Bross B, Oudin S, Helle P, Marpe D, Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. In: *Proc. SPIE.* 8499, Applications of Digital Image Processing XXXV, no. 84990R, Oct. 2012
10. De Forni R, Taubman D (2005) On the benefits of leaf merging in quad-tree motion models. In: IEEE international conference on image processing, pp II–858, IEEE, 2005
11. Han WJ, Min J, Kim IK, Alshina E, Alshin A, Lee T, Chen J, Seregin V, Lee S, Hong YM, Cheon MS, Shlyakhov N, McCann K, Davies T, Park JH (2010) Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools. IEEE Trans Circuits Syst Video Technol 20(12):1709–1720
12. Helle P, Oudin S, Bross B,Marpe D, Bici M, Ugur K, Jung J, Clare G,Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. IEEE Trans Circuits Syst Video Technol 22(12):1720–1731
13. JCT-VC (2014) Subversion repository for the HEVC test model reference software. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware
14. Jung J, Bross B (2011) CE9: Summary report for CE9 on motion vector coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D149, Daegu, Jan. 2011
15. Jung J, Onno P, Huang YW (2011) CE1: Summary report of core experiment 1 on motion data storage reduction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F021, Torino, July 2011
16. Kamp S, Ballé J, Wien M (2009) Multihypothesis prediction using decoder side-motion vector derivation in inter-frame video coding. In: *Proc. SPIE.* 7257, Visual Communications and Image Processing 2009, no. 725704, Jan. 2009
17. Ugur K, Alshin A, Alshina E, Bossen F, HanW, Park J, Lainema J (2013) Motion compensated prediction and interpolation filter design in H.265/HEVC. IEEE J Sel Top Signal Process 7(6): 946–956
18. Ugur K, Lainema J, Hallapuro A (2011) High precision bi-directional averaging, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D321, Daegu, Jan. 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00150

19. Laroche G, Jung J, Pesquet-Popescu B (2008) RD optimized coding for motion vector predictor selection. IEEE Trans Circuits Syst Video Technol 18(9):1247–1257
20. Li B, Xu J (2011) Parsing robustness in high efficiency video coding-analysis and improvement. In: IEEE visual communications and image processing (VCIP), pp 1–4, 2011
21. Li B, Xu J, Li H (2011) Non-CE9/Non-CE13: Simplification of adding new merge candidates, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G397, Geneva, Nov. 2011
22. Mathew R, Taubman DS (2010) Quad-tree motion modeling with leaf merging. IEEE Trans Circuits Syst Video Technol 20(10):1331–1345
23. Oudin S, Helle P, Stegemann J, Bartnik C, Bross B, Marpe D, Schwarz H, Wiegand T (2011) Block merging for quadtree-based video coding. In: IEEE international conference on multimedia and expo, pp 1–6, IEEE, 2011
24. Pu W, Chen J, Karczewicz M, Kim WS, Sole J, Guo L (2013) High precision weighted prediction for HEVC range extension, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-O0235, Geneva, Oct.-Nov. 2013
25. Sugio T, Nishi T (2011) Parsing robustness for merge/AMVP, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F470, Torino, July 2011
26. Tourapis AM, Wu F, Li S (2005) Direct mode coding for bipredictive slices in the H.264 standard. IEEE Trans Circuits Syst Video Technol 15(1):119–126
27. Wiegand T, Sullivan GJ, Bjøntegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576
28. Zhou M (2012) AHG10: Configurable and CU-group level parallel merge/skip, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0082, San Jose, Feb. 2012

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00151

Chapter 6
HEVC Transform and Quantization

Madhukar Budagavi, Arild Fuldseth, and Gisle Bjøntegaard

Abstract This chapter provides an overview of the transform and quantization design in HEVC. HEVC specifies two-dimensional transforms of various sizes from 4×4 to 32×32 that are finite precision approximations to the discrete cosine transform (DCT). In addition, HEVC also specifies an alternate 4×4 integer transform based on the discrete sine transform (DST) for use with 4×4 luma Intra prediction residual blocks. During the transform design, special care was taken to allow implementation friendliness, including limited bit depth, preservation of symmetry properties, embedded structure and basis vectors having almost equal norm. The HEVC quantizer design is similar to that of H.264/AVC where a quantization parameter (QP) in the range of 0–51 (for 8-bit video sequences) is mapped to a quantizer step size that doubles each time the QP value increases by 6. A key difference, however, is that the transform basis norm correction factors incorporated into the descaling matrices of H.264/AVC are no longer needed in HEVC simplifying the quantizer design. A QP value can be transmitted (in the form of delta QP) for a quantization group as small as 8×8 samples for rate control and perceptual quantization purposes. The QP predictor used for calculating the delta QP uses a combination of left, above and previous QP values. HEVC also supports frequency-dependent quantization by using quantization matrices for all transform block sizes. This chapter also provides an overview of the three special coding modes in HEVC (I_PCM mode, lossless mode, and transform skip mode) that modify the transform and quantization process by either skipping the transform or by skipping both transform and quantization.

M. Budagavi (✉)
Texas Instruments Inc., Dallas, TX, USA
e-mail: madhu072@yahoo.com

A. Fuldseth • G. Bjøntegaard
Cisco Systems Norway, 1366 Lysaker, Norway
e-mail: arild.fuldseth@cisco.com; gbjonteg@cisco.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_6
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00152

6.1 Introduction

In the block-based hybrid video coding approach, transforms are applied to the residual signal resulting from inter- or intra-picture prediction as shown in Fig. 6.1. At the encoder, the residual signal of a picture is divided into square blocks of size $N \times N$ where $N = 2^M$ and M is an integer. Each residual block (U) is then input to a two-dimensional $N \times N$ forward transform. The two-dimensional transform can be implemented as a separable transform by applying an N-point one-dimensional transform to each row and each column separately. The resulting $N \times N$ transform coefficients (*coeff*) are then subject to quantization (which is equivalent to division by quantization step size *Qstep* and subsequent rounding) to obtain quantized transform coefficients (*level*). At the decoder, the quantized transform coefficients are then de-quantized (which is equivalent to multiplication by *Qstep*). Finally, a two-dimensional $N \times N$ separable inverse transform is applied to the de-quantized transform coefficients ($coeff_Q$) resulting in a residual block of quantized samples which is then added to the intra- or inter-prediction samples to obtain the reconstructed block.

Typically, the forward- and inverse transform matrices are transposes of each other and are designed to achieve near lossless reconstruction of the input residual block when concatenated without the intermediate quantization and de-quantization steps.

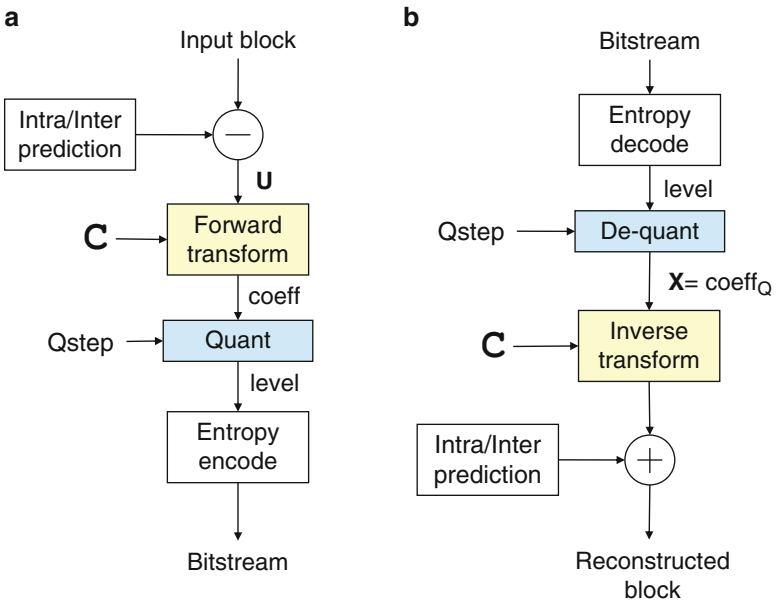


Fig. 6.1 Block-based hybrid video coding. (**a**) Encoder, (**b**) Decoder. **C** is the transform matrix and *Qstep* is the quantization step size. Reproduced with permission from [6]. © IEEE 2013

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00153

In video coding standards such as HEVC, the de-quantization process and inverse transforms are specified, while the forward transforms and quantization process are chosen by the implementer (subject to constraints on the bitstream).

This chapter is organized as follows. Section 6.2 describes the two transform types used in HEVC: the *core* transform based on the discrete cosine transform and the *alternate* transform based on the discrete sine transform. Design principles used to develop the transform are also highlighted to provide insight into the transform design process which considered both coding efficiency and complexity. In Sect. 6.3, the HEVC quantization process is described. Topics covered in this section include the actual quantization and de-quantization steps, quantization matrices, and quantization parameter derivation. Section 6.4 provides an overview of the three special coding modes in HEVC (I_PCM mode, Lossless mode, and Transform skip mode) that modify the transform and quantization process by either skipping the transform or by skipping both transform and quantization. Sections 6.5 and 6.6 provide complexity analysis and coding performance results respectively.

6.2 HEVC Transform[1]

The HEVC standard [16] specifies core transform matrices of size 4×4, 8×8, 16×16 and 32×32 to be used for two-dimensional transforms in the context of block-based motion-compensated video compression. Multiple transform sizes improve compression performance, but also increase the implementation complexity. Hence a careful design of the core transforms is needed.

HEVC specifies two-dimensional core transforms that are finite precision approximations to the inverse discrete cosine transform (IDCT) for all transform sizes. Note that because of the approximations, the HEVC core transforms are not the IDCT. The fact that an IDCT is not used does not necessarily make the HEVC core transforms imperfect. In fact, the finite precision approximations are desirable as explained in the next two paragraphs. The main purpose of the transform is to de-correlate the input residual block. The optimal de-correlating transform is the Karhunen–Loeve transform (KLT) [22] and not necessarily the DCT. This is especially true for the coding of 4×4 luma intra-prediction residual blocks where HEVC specifies an alternate 4×4 integer transform based on the discrete sine transform (DST) [24]. Note that only the inverse transforms are specified in the HEVC standard and the forward transforms are not. So an encoder may get additional coding efficiency benefits by using the actual inverse rather than the transpose of the inverse transform.

[1]Portions of this section are © 2013 IEEE. Reprinted, with permission, from M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, M. Sadafale, "Core Transform Design in the High Efficiency Video Coding (HEVC) Standard," IEEE Journal of Selected Topics in Signal Processing, December 2013.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00154

In the H.261, MPEG-1, H.262/MPEG-2, and H.263 video coding standards, an 8-point IDCT was specified with infinite precision. To ensure interoperability and to minimize drift between encoder and decoder implementations using finite precision, two features were included in the standards. First, block-level periodic intra refresh was mandatory. Second, a conformance test for the accuracy of the IDCT using a pseudo-random test pattern was specified.

In the H.264/MPEG-4 Advanced Video Coding (AVC) standard [15], the problem of encoder–decoder drift was solved by specifying integer valued 4×4 and 8×8 transform matrices. The transforms were designed as approximations to the IDCT with emphasis on minimizing the number of arithmetic operations. These transforms had large variations of the norm of the basis vectors. As a consequence of this, non-flat default de-quantization matrices were specified to compensate for the different norms of the basis vectors [20].

During the development of HEVC, several different approximations of the IDCT were studied for the core transform. The first version of the HEVC Test Model HM1 used the H.264/AVC transforms for 4×4 and 8×8 blocks and integer approximation of Chen's fast IDCT [7] for 16×16 and 32×32 blocks. The HM1 inverse transforms had the following characteristics [23, 28]:

- Non-flat de-quantization matrices for all transform sizes: While acceptable for small transform sizes, the implementation cost of using de-quantization matrices for larger transforms is high because of larger block sizes,
- Different architectures for different transform sizes: This leads to increased area since hardware sharing across different transform sizes is difficult,
- A 20-bit transpose buffer used for storing intermediate results after the first transform stage in 2D transform: An increased transpose buffer size leads to larger memory and memory bandwidth. In hardware, the transpose buffer area can be significant and comparable to transform logic area [30],
- Full factorization architecture requiring cascaded multipliers and intermediate rounding for 16- and 32-point transforms: This increases data path dependencies and impacts parallel processing performance. It also leads to increased bit width for multipliers and accumulators (32 bits and 64 bits respectively in software). In hardware, in addition to area increase, it also leads to increased circuit delay thereby limiting the maximum frequency at which the inverse transform block can operate.

To address the complexity concerns of the HM1 transforms, a matrix multiplication based core transform was proposed in [10] and eventually adopted as the HEVC core transform. The design goal was to develop a transform that was efficient to implement in both software on SIMD machines and in hardware. Alternative proposals to the HEVC core transform design can be found in [1, 9, 17].

The HEVC core transform matrices were designed to have the following properties [10]:

- Closeness to the IDCT
- Almost orthogonal basis vectors

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00155

- Almost equal norm of all basis vectors
- Same symmetry properties as the IDCT basis vectors
- Smaller transform matrices are embedded in larger transform matrices
- Eight-bit representation of transform matrix elements
- Sixteen-bit transpose buffer
- Multipliers can be represented using 16 bits or less with no cascaded multiplications or intermediate rounding
- Accumulators can be implemented using less than 32 bits

6.2.1 Discrete Cosine Transform

The N transform coefficients v_i of an N-point 1D DCT applied to the input samples u_i can be expressed as

$$v_i = \sum_{j=0}^{N-1} u_j c_{ij} \tag{6.1}$$

where $i = 0, \ldots, N-1$. Elements c_{ij} of the DCT transform matrix C are defined as

$$c_{ij} = \frac{P}{\sqrt{N}} \cos\left[\frac{\pi}{N}\left(j + \frac{1}{2}\right) i\right] \tag{6.2}$$

where $i,j = 0, \ldots, N-1$ and where P is equal to 1 and $\sqrt{2}$ for $i = 0$ and $i > 0$, respectively. Furthermore, the basis vectors $\mathbf{c}_i$ of the DCT are defined as $\mathbf{c}_i = [c_{i0}, \ldots, c_{i(N-1)}]^T$ where $i = 0, \ldots, N-1$.

The DCT has several properties that are considered useful both for compression efficiency and for efficient implementation [22].

1. The basis vectors are orthogonal, i.e. $\mathbf{c}_i^T \mathbf{c}_j = 0$ for $i \neq j$. This property is desirable for compression efficiency by achieving transform coefficients that are uncorrelated.
2. The basis vectors of the DCT have been shown to provide good energy compaction which is also desirable for compression efficiency.
3. The basis vectors of the DCT have equal norm, i.e. $\mathbf{c}_i^T \mathbf{c}_i = 1$ for $i = 0, \ldots, N-1$. This property is desirable for simplifying the quantization/de-quantization process. Assuming that equal frequency-weighting of the quantization error is desired, equal norm of the basis vectors eliminates the need for quantization/de-quantization matrices.
4. Let $N = 2^M$. The elements of a DCT matrix of size $2^M \times 2^M$ is a subset of the elements of a DCT matrix of size $2^{M+1} \times 2^{M+1}$. More specifically, the basis vectors of the smaller matrix is equal to the first half of the even basis vectors of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00156

the larger matrix. This property is useful to reduce implementation costs as the same multipliers can be reused for various transform sizes.

5. The DCT matrix can be specified by using a small number of unique elements. By examining the elements c_{ij} of (6.2) it can be shown that the number of unique elements in a DCT matrix of size $2^M \times 2^M$ is equal to $2^M - 1$. As further elaborated in Sect. 6.2.4, this is particularly advantageous in hardware implementations.
6. The even basis vectors of the DCT are symmetric, while the odd basis vectors are anti-symmetric. This property is useful to reduce the number of arithmetic operations.
7. The coefficients of a DCT matrix have certain trigonometric relationships that allows for a reduction of the number of arithmetic operations beyond what is possible by exploiting the (anti-)symmetry properties. These properties can be utilized to implement fast algorithms such as the Chen's fast factorization [7].

6.2.2 Finite Precision DCT Approximations

The core transform matrices of HEVC are finite precision approximations of the DCT matrix. The benefit of using finite precision in a video coding standard is that the approximation to the real-valued DCT matrix is specified in the standard rather than being implementation dependent. This avoids encoder–decoder mismatch and drift caused by manufacturers implementing the IDCT with slightly different floating point representations. On the other hand, a disadvantage of using approximate matrix elements is that some of the properties of the DCT discussed in Sect. 6.2.1 may not be satisfied anymore. More specifically, there is a trade-off between the computational cost associated with using high bit-depth for the matrix elements and the degree to which some of the conditions of Sect. 6.2.1 are satisfied.

A straightforward way of determining integer approximations to the DCT matrix elements is to scale each matrix element with some large number (typically between 2^5 and 2^{16}) and then round to the closest integer. However, this approach does not necessarily result in the best compression performance. As shown in Sect. 6.2.3, for a given bit-depth of the matrix elements, a different strategy for approximating the DCT matrix elements results in a different trade-off between some of the properties of Sect. 6.2.1.

6.2.3 HEVC Core Transform Design Principles

The DCT approximations used for the core transforms of HEVC were chosen according to the following principles. First, properties 4–6 of Sect. 6.2.1 were satisfied without any compromise. This choice ensures that several implementation friendly aspects of the DCT are preserved. Second, for properties 1–3 and 7 of Sect. 6.2.1, there were trade-offs between the number of bits used to represent each matrix element and the degree by which each of the properties were satisfied.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00157

Table 6.1 Comparison of transform design methods

	HEVC core transforms	Scaling and rounding
Orthogonality	$o_{ij} < 0.0029$	$o_{ij} < 0.0037$
Closeness to DCT	$m_{ij} < 0.0213$	$m_{ij} < 0.0077$
Norm measure	$n_i < 0.0014$	$n_i < 0.0109$

To measure the degree of approximation for properties 1–3 of Sect. 6.2.1, the following measures are defined for an integer N-point DCT approximation with scaled matrix elements equal to d_{ij} and basis vectors equal to $\mathbf{d}_i = [d_{i0}, \ldots, d_{i(N-1)}]^T$ where $i = 0, \ldots, N-1$.

1. Orthogonality measure: $o_{ij} = \mathbf{d}_i^T \mathbf{d}_j / \mathbf{d}_0^T \mathbf{d}_0, i \neq j$
2. Closeness to DCT measure: $m_{ij} = |\alpha c_{ij} - d_{ij}| / d_{00}$
3. Norm measure: $n_i = |1 - \mathbf{d}_i^T \mathbf{d}_i / \mathbf{d}_0^T \mathbf{d}_0|$

where $i, j = 0, \ldots, N-1$, c_{ij} are the DCT matrix elements of (6.2), and the scale factor α is defined as $d_{00} N^{1/2}$.

As a result of careful investigation, it was decided to represent each matrix coefficient with 8 bit (including sign bit), and to choose the elements of the first basis vector to be equal to 64 (i.e. $d_{0j} = 64, j = 0, \ldots, N-1$). Note that this results in a scale factor of $2^{6+M/2}$ for the HEVC transform matrix when compared to the orthonormal DCT. The remaining matrix elements were hand-tuned (within the constraints of properties 4–6 of Sect. 6.2.1) to achieve a good balance between properties 1–3 of Sect. 6.2.1. The hand-tuning was performed as follows. First, the real-valued scaled DCT matrix elements, αc_{ij}, were derived. Next, for each unique number in the resulting matrices, each integer value in the interval $[-1.5, 1.5]$ around αc_{ij} was examined and the resulting values of o_{ij}, m_{ij}, and n_i were calculated. Since there are only 31 unique numbers in the transform matrices (see Sect. 6.2.4), various permutations can be examined systematically (although not exhaustively). The final integer matrix elements were chosen to give a good compromise between all measures o_{ij}, m_{ij}, and n_i. The resulting worst case values of o_{ij}, m_{ij}, and n_i are shown in the second column of Table 6.1. The norm was considered to be sufficiently close to 1 (i.e. the norm measure n_i is sufficiently close to 0) to justify not using a non-flat default de-quantization matrix in HEVC (i.e. all transform coefficients scaled equally).

For comparison purposes, the resulting measures when multiplying the real-valued DCT matrix elements with $2^{6+M/2}$ and rounding to the closest integer are listed in the third column of Table 6.1. As can be seen from the table, although the matrix elements of the HEVC transforms are farther from the scaled DCT matrix elements, they have better orthogonality and norm properties.

Finally, by using only 8 bit representation, property 7 of Sect. 6.2.1 (trigonometric relationship between matrix elements) was not easily preserved. The authors are not aware of any trigonometric property of the HEVC core transforms that can be utilized to reduce the number of arithmetic operations below those required when using the (anti-) symmetry properties.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00158

6.2.4 Basis Vectors of the HEVC Core Transforms

The left half of the 32×32 matrix specifying the 32-point forward transform is shown in Fig. 6.2. The right half can be derived by using the (anti-) symmetry properties of the basis vectors (property 6 of Sect. 6.2.1). The inverse transform matrix of HEVC is defined as the transpose of the matrix resulting from the figure. The 32×32 matrix contains up to 31 unique numbers as follows.

$$d^{32}_{i,0}, i = 1, \ldots, 31 = \begin{Bmatrix} 90, 90, 90, 89, 88, 87, 85, 83, 82, 80, 78, 75, 73, 70, 67, 64, \\ 61, 57, 54, 50, 46, 43, 38, 36, 31, 25, 22, 18, 13, 9, 4 \end{Bmatrix} \tag{6.3}$$

These unique numbers are elements 1–31 of the first column of the forward transform matrix. Note that although the number 90 occurs three times, this is by accident and not generally true. The unique numbers property was used in [26] to enable 25 % area reduction for hardware designs with practical throughput.

Furthermore, the coefficients d^N_{ij} of the smaller transform matrices ($N = 4, 8, 16$) can be derived from the coefficients d^{32}_{ij} of the 32×32 transform matrix as:

$$d^N_{ij} = d^{32}_{i(32/N),j}, i, j = 0, \ldots, N-1 \tag{6.4}$$

Let $\mathbf{D}_4$ denote the 4×4 transform matrix. By using (6.4) and Fig. 6.2, $\mathbf{D}_4$ can be obtained as:

$$\mathbf{D}_4 = \begin{bmatrix} d^{32}_{0,0} & d^{32}_{0,1} & d^{32}_{0,2} & d^{32}_{0,3} \\ d^{32}_{8,0} & d^{32}_{8,1} & d^{32}_{8,2} & d^{32}_{8,3} \\ d^{32}_{16,0} & d^{32}_{16,1} & d^{32}_{16,2} & d^{32}_{16,3} \\ d^{32}_{24,0} & d^{32}_{24,1} & d^{32}_{24,2} & d^{32}_{24,3} \end{bmatrix} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix}$$

The 8×8 transform matrix $\mathbf{D}_8$ and the 16×16 transform matrix $\mathbf{D}_{16}$ can be similarly obtained from the 32×32 transform matrix as shown in Fig. 6.2 where different colors are used to highlight the embedded 16×16, 8×8 and 4×4 forward transform matrices. This property allows for different transform sizes to be implemented using the same architecture thereby facilitating hardware sharing [6].

Note that from the unique numbers property of (6.3) and the (anti-)symmetry properties, $\mathbf{D}_4$ is also equal to:

$$\mathbf{D}_4 = \begin{bmatrix} d^{32}_{16,0} & d^{32}_{16,0} & d^{32}_{16,0} & d^{32}_{16,0} \\ d^{32}_{8,0} & d^{32}_{24,0} & -d^{32}_{24,0} & -d^{32}_{8,0} \\ d^{32}_{16,0} & -d^{32}_{16,0} & -d^{32}_{16,0} & d^{32}_{16,0} \\ d^{32}_{24,0} & -d^{32}_{8,0} & d^{32}_{8,0} & -d^{32}_{24,0} \end{bmatrix} \tag{6.5}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00159

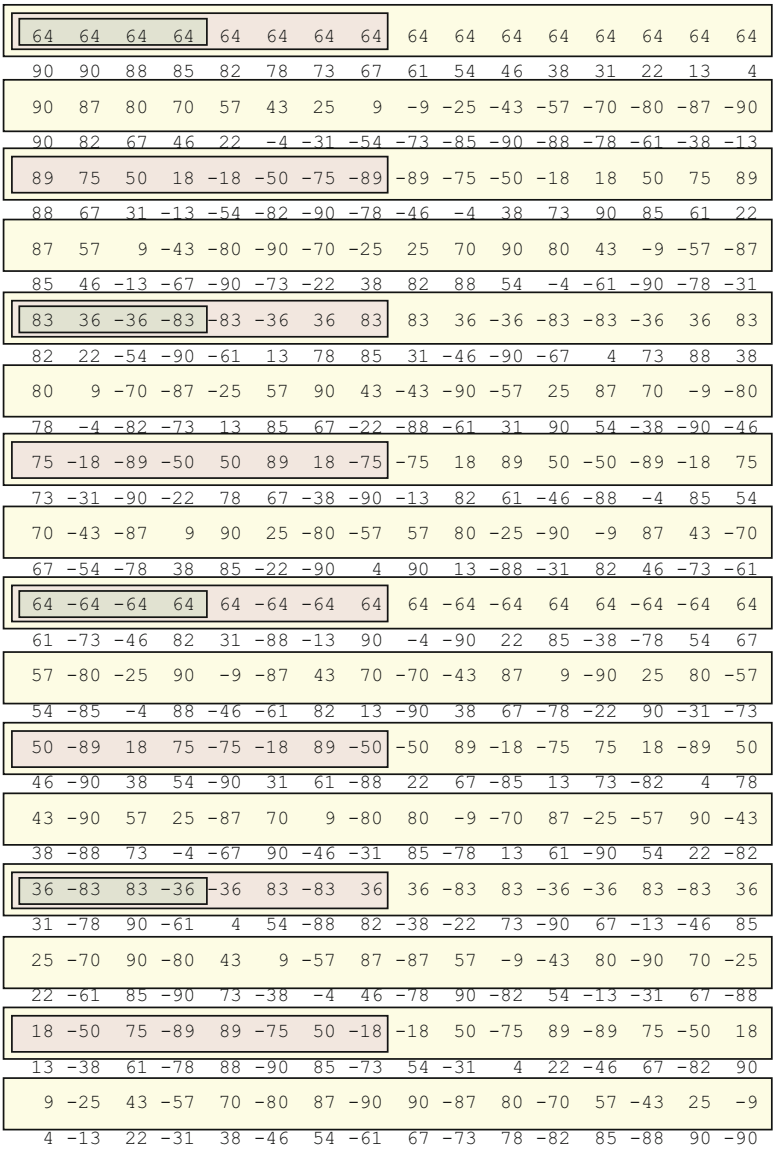


Fig. 6.2 Left half of the 32×32 matrix specifying the 32-point forward transform. Embedded 4-point (*green shading*), 8-point (*pink shading*) and 16-point (*yellow shading*) forward transform matrices are also shown in the figure. Reproduced with permission from [6]. © IEEE 2013

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00160

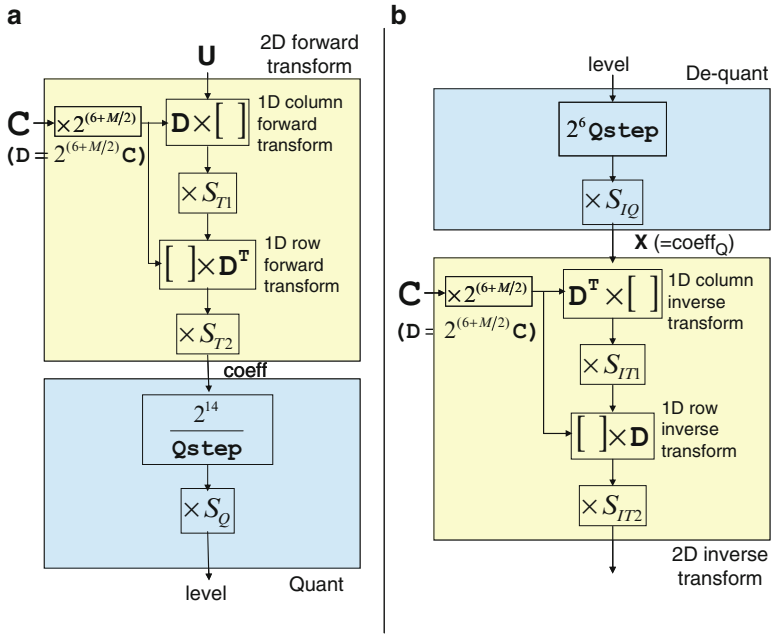


Fig. 6.3 Additional scale factors $S_{T1}, S_{T2}, S_{IT1}, S_{IT2}, S_Q, S_{IQ}$ required to implement HEVC integer transform and quantization. (**a**) Forward transform and quantization, (**b**) inverse transform and quantization. The 2D forward and inverse transform are implemented as separable 1D column and row transforms. **C** is the orthonormal DCT matrix. **D** is the scaled approximation of the DCT matrix. $M = \log_2(N)$ where N is the transform size. Reproduced with permission from [6]. © IEEE 2013

6.2.5 *Intermediate Scaling*

Since the HEVC matrices are scaled by $2^{(6+M/2)}$ compared to an orthonormal DCT transform, and in order to preserve the norm of the residual block through the forward and inverse two-dimensional transforms, additional scale factors—$S_{T1}, S_{T2}, S_{IT1}, S_{IT2}$ —need to be applied as shown in Fig. 6.3. Note that Fig. 6.3 is basically a fixed point implementation of the transform and quantization in Fig. 6.1. While the HEVC standard specifies the scale factors of the inverse transform (i.e. S_{IT1}, S_{IT2}), the HEVC reference software also specifies corresponding scale factors for the forward transform (i.e. S_{T1}, S_{T2}). The scale factors were chosen with the following constraints:

1. All scale factors shall be a power of two to allow the scaling to be implemented as a right shift.
2. Assuming full range of the input residual block (e.g. a DC block with all samples having maximum amplitude), the bit depth after each transform stage shall be equal to 16 bits (including the sign bit). This was considered a reasonable trade-off between accuracy and implementation costs.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00161

3. Since the HEVC matrices are scaled by $2^{(6+M/2)}$, cascading of the two-dimensional forward and inverse transform will results in a scaling of $2^{(6+M/2)}$ for each of the 1D row forward transform, the 1D column forward transform, the 1D column inverse transform, and the 1D row inverse transform. Consequently to preserve the norm through the two-dimensional forward and inverse transforms, the product of all scale factors shall be equal to $(1/2^{(6+M/2)})^4 = 2^{-24}2^{-2M}$.

The process of selecting the forward transform scale factors is illustrated using the 4×4 forward transform as an example in Fig. 6.4. When video has a bit depth of B bits, the residual will be in the range of $[-2^B + 1, 2^B - 1]$ requiring $(B + 1)$ bits to represent it. In the following worst case bit-depth analysis we will assume a residual block with all samples having maximum amplitude equal to -2^B as input to the first stage of the forward transform. We believe this is a reasonable assumption since all basis vectors have almost the same norm. Note also that we are using -2^B instead of $-2^B + 1$ or $2^B - 1$ in the worst case analysis since it is a power of 2. The scale factor derivation becomes simpler assuming input to be -2^B (which still fits within $(B + 1)$ bits) since all the scale factors are a power of 2. For this worst case input block, the maximum value of an output sample will be $-2^B \times N \times 64$. This corresponds to the dot product of the first basis vector (of length N with all values equal to 64) with an input vector consisting of values equal to -2^B. Therefore, with $N = 2^M$, for the output to fit within 16 bits (i.e., maximum value of -2^{15}) a scaling of $1/(2^B \times 2^M \times 2^6 \times 2^{-15})$ is required. Consequently, the scale factor after the first transform stage is chosen as $S_{T1} = 2^{-(B+M-9)}$.

The second stage of the forward transform consists of multiplication of the result of the first transform stage with $\mathbf{D}_4^T$. The input into the second stage of the forward transform is the output from the first stage which is a matrix with all elements in the first row having a value of -2^{15}. All other elements will be zero as shown in Fig. 6.4. The output of multiplication with $\mathbf{D}_4^T$ will be a matrix with only a DC value equal to $-2^{15} \times 2^M \times 2^6$ and all remaining values equal to 0. This implies that the scaling required after the second stage of transform is $S_{T2} = 2^{-(M+6)}$ in order for the output to fit within 16 bits.

The first stage of the inverse transform consists of multiplication of the result of the forward transform with $\mathbf{D}_4^T$. In our example, the input into the first stage of the inverse transform is the output matrix from the forward transform which is a matrix with only the DC element equal to -2^{15}. The output of multiplication with $\mathbf{D}_4^T$ will be a matrix with first column elements equal to $-2^{15} \times 2^6$. Consequently, the scaling required after the first stage of the inverse transform for the output to fit within 16 bits is $S_{IT1} = 2^{-6}$.

The second stage of the inverse transform consists of multiplication of the result of the first stage of the inverse transform with $\mathbf{D}_4$. The input into the second stage of the inverse transform is the output matrix from the first stage of inverse transform which is a matrix with first column elements equal to -2^{15}. The output of multiplication with $\mathbf{D}_4$ will be a matrix with all elements equal to $-2^{15} \times 2^6$. So the scaling required after the second stage of inverse transform to get the output values into the original range of $[-2^B, 2^B - 1]$ is $S_{IT2} = 2^{-(21-B)}$.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00162

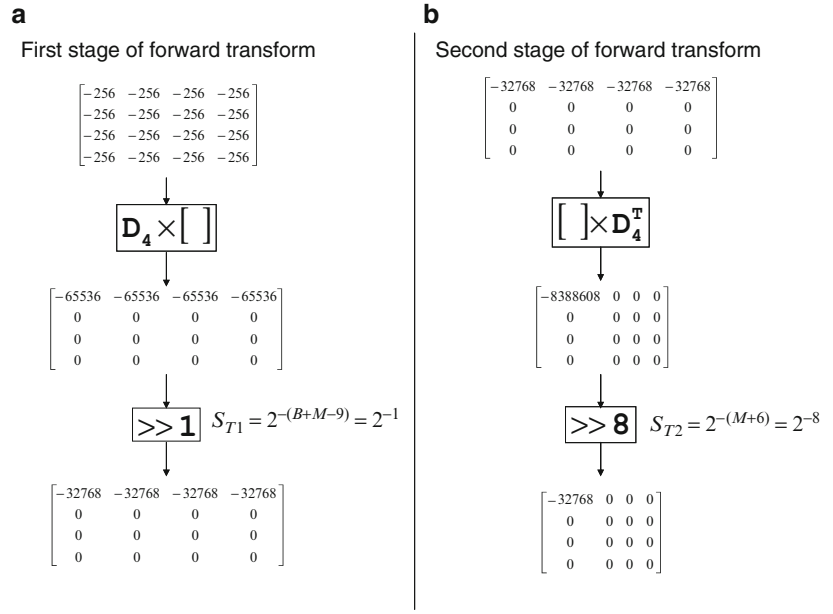


Fig. 6.4 Intermediate scaling factor determination for the forward transform so that the intermediate and output values fit within 16-bits. B is video bit depth and $M = \log_2(N)$ where N is the transform size. Worst case bit-depth analysis is done assuming a residual block with all samples having maximum amplitude equal to -2^B (where $B = 8$ is the video bit depth), as input to the first stage of the forward transform. (**a**) First stage of the forward transform, (**b**) Second stage of the forward transform. Reproduced with permission from [6]. © IEEE 2013

In summary the constraints imposed in this section result in the following scale factors after different transform stages:

- After the first forward transform stage: $S_{T1} = 2^{-(B+M-9)}$
- After the second forward transform stage: $S_{T2} = 2^{-(M+6)}$
- After the first inverse transform stage: $S_{IT1} = 2^{-6}$
- After the second inverse transform stage: $S_{IT2} = 2^{-(21-B)}$

where B is the bit depth of the input/output signal (e.g. 8 bit) and $M = \log_2(N)$.

Without quantization/de-quantization, this choice of scale factors ensures a bit depth of 16 bit after all transform stages. However, quantization errors introduced by the quantization/de-quantization process might increase the dynamic range before each inverse transform stage to more than 16 bit. For example, consider the situation where $B = 8$ and all input samples to the forward transform are equal to 255. In this case, the output of the forward transform will be a DC coefficient with value equal to $255 << 7 = 32640$. For high QP values and with a quantizer rounding upwards, the input to each inverse transform stage can easily exceed the allowed 16 bit dynamic range of $[-32768,\ 32767]$. While clipping to 16 bit range was considered trivial after the de-quantizer, it was considered undesirable after the first inverse transform stage. In order to allow for quantization error of some reasonable magnitude and at

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00163

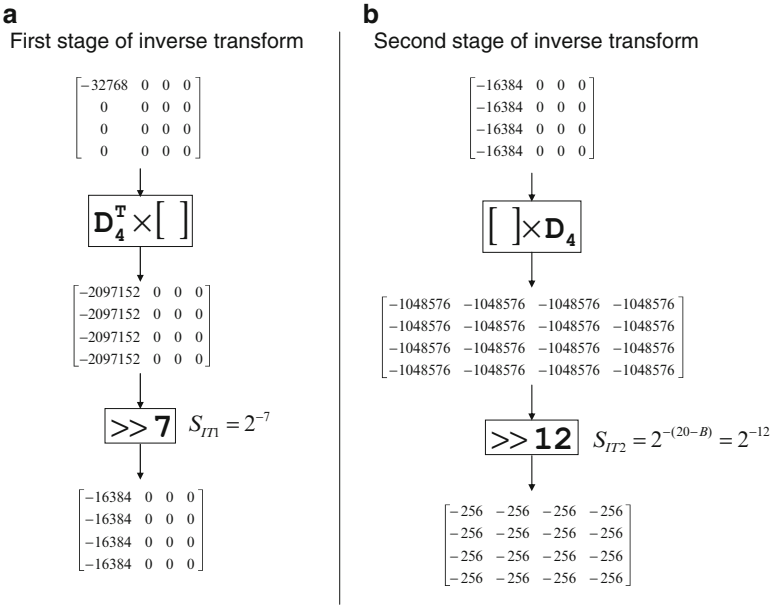


Fig. 6.5 Use of the inverse transform scale factors assuming the input to be the final output of Fig. 6.4. Video bit depth $B = 8$ (**a**) First stage of the inverse transform, (**b**) Second stage of the inverse transform. Reproduced with permission from [6]. © IEEE 2013

the same time limit the dynamic range between the two inverse transform stages to 16 bits, the choice of scale factors for the inverse transform was finally modified as follows[2]:

- After the first inverse transform stage: $S_{IT1} = 2^{-7}$
- After the second inverse transform stage: $S_{IT2} = 2^{-(20-B)}$

The use of the inverse transform scale factors is illustrated in Fig. 6.5 using the 4×4 inverse transform as an example assuming the input to be the final output of Fig. 6.4.

Tables 6.2 and 6.3 summarize the different scaling factors of the forward and inverse transform, respectively, when compared to the orthonormal DCT.

The HEVC specification specifies an offset value to be added before scaling to carry out rounding. This offset value is equal to the scale factor divided by 2. The offset is not explicitly shown in Figs. 6.3, 6.4, and 6.5.

[2]Note that in the final HEVC specification [16], a clipping operation is introduced after the first inverse transform stage, mainly to allow for random quantization that could be used to create "evil" bitstreams used for stress testing video decoders. With the clipping introduced, the modification to the inverse transform scale factors is not necessary but has been retained in the HEVC specification and Test Model software for maturity reasons.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00164

Table 6.2 Scaling in different stages for the 2D forward transform

	Scale factor
First forward transform stage	$2^{(6+M/2)}$
After the first forward transform stage (S_{TI})	$2^{-(B+M-9)}$
Second forward transform stage	$2^{(6+M/2)}$
After the second forward transform stage (S_{T2})	$2^{-(M+6)}$
Total scaling for the forward transform	$2^{(15-B-M)}$

Table 6.3 Scaling in different stages for the 2D inverse transform

	Scale factor
First inverse transform stage	$2^{(6+M/2)}$
After the first inverse transform stage (S_{ITI})	2^{-7}
Second inverse transform stage	$2^{(6+M/2)}$
After the second inverse transform stage (S_{IT2})	$2^{-(20-B)}$
Total scaling for the inverse transform	$2^{-(15-B-M)}$

Finally, two useful consequences of using 8-bit coefficients and limiting the bit-depth of the intermediate data to 16 bit is that all multiplications can be represented with multipliers having 16 bits or less and that the accumulators before right shift can be implemented with less than 32 bits for all transform stages.

Note also a relevant analysis in [18] that studies the dynamic range of the HEVC inverse transform and provides additional information on the bit depth limits of the intermediate data in the inverse transform.

6.2.6 HEVC Alternate 4 × 4 Transform

The alternate transform is applied to 4×4 Luma intra-prediction residual blocks. The forward transform matrix is given by:

$$\mathbf{A}_4 = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}$$

The inverse transform matrix is $\mathbf{A}_4^T$. Elements a_{ij} of the alternate transform matrix $\mathbf{A}_4$ are a fixed point representation of Type-7 discrete sine transform (DST) obtained as follows:

$$a_{ij} = \text{round}\left(128 * \frac{2}{\sqrt{2N+1}} \sin\left(\frac{(2i+1)(j+1)\pi}{2N+1}\right)\right)$$

The intermediate scaling and quantization/de-quantization used for the alternate transform is the same as that for the core transform.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00165

The alternate transform provides around 1 % bit-rate reduction while coding intra pictures [25]. In intra-picture prediction, a block is predicted from left and/or top neighboring samples. The prediction quality is better near the left and/or top boundary resulting in an intra-prediction residual that tends to have lower amplitude near the boundary samples and higher amplitudes away from the boundary samples. The DST basis functions are better than the DCT basis functions in modeling this spatial characteristic of the intra prediction residual. This can be seen from the first row (basis function) of the alternate transform matrix which increases from left to right as opposed to the DCT transform matrix that has a flat first row. A theoretical analysis of the optimality of DST for intra-prediction residual is provided in [25].

During the course of the development of HEVC, alternate transforms for transform block sizes of 8×8 and higher were also studied. However, only the 4×4 alternate transform was adopted in HEVC since the additional coding gain from using the larger alternate transforms was not significant (also, their complexity is higher since there is no symmetry in the transform matrix and a full matrix multiplication is needed to implement them for transform sizes 8×8 and larger).

6.3 Quantization and De-quantization

Quantization consists of division by a quantization step size (*Qstep*) and subsequent rounding while inverse quantization consists of multiplication by the quantization step size. Here, *Qstep* refers to the equivalent step size for an orthonormal transform, i.e. without the scaling factors of Tables 6.2 and 6.3. Similar to H.264/AVC [27], a quantization parameter (*QP*) is used to determine the quantization step size in HEVC. *QP* can take 52 values from 0 to 51 for 8-bit video sequences. An increase of 1 in *QP* means an increase of the quantization step size by approximately 12 % (i.e., $2^{1/6}$). An increase of 6 leads to an increase in the quantization step size by a factor of 2. In addition to specifying the *relative* difference between the step-sizes of two consecutive *QP* values, there is also a need to define the *absolute* step-size associated with the range of *QP* values. This was done by selecting $Qstep = 1$ for $QP = 4$.

The resulting relationship between *QP* and the equivalent quantization step size for an orthonormal transform is now given by:

$$Qstep(QP) = \left(2^{1/6}\right)^{QP-4} \tag{6.6}$$

Figure 6.6 shows how the quantization step size increases non-linearly with *QP*. Equation (6.6) can be also be expressed as:

$$Qstep(QP) = G_{QP\%6} << \frac{QP}{6} \tag{6.7}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00166

where

$$\mathbf{G} = [G_0, G_1, G_2, G_3, G_4, G_5]^T = \left[2^{-4/6}, 2^{-3/6}, 2^{-2/6}, 2^{-1/6}, 2^0, 2^{1/6}\right]^T$$

The fixed point approximation of (6.7) in HEVC is given by

$$g_{QP\%6} = \text{round}\left(2^6 \times G_{QP\%6}\right)$$

This results in

$$\mathbf{g} = [g_0, g_1, g_2, g_3, g_4, g_5]^T = [40, 45, 51, 57, 64, 72]^T$$

HEVC supports frequency-dependent quantization by using quantization matrices for all transform block sizes. Let $W[x][y]$ denote the quantization matrix weight for the transform coefficients at location (x, y) in a transform block. A value of $W[x][y] = 1$ indicates that there is no weighting. The fixed point representation of $W[x][y]$ is given by:

$$w\,[x]\,[y] = \text{round}\,(16 \times W\,[x]\,[y])$$

where $w[x][y]$ is represented using 8-bit values.

For a quantizer output, $level[x][y]$, the de-quantizer is specified in the HEVC standard as

$$coeff_{\,Q}\,[x]\,[y] = \left(\left(level\,[x]\,[y] \times w\,[x]\,[y] \times \left(g_{QP\%6} << \frac{QP}{6}\right)\right) + offset_{1Q}\right) >> shift1 \tag{6.8}$$

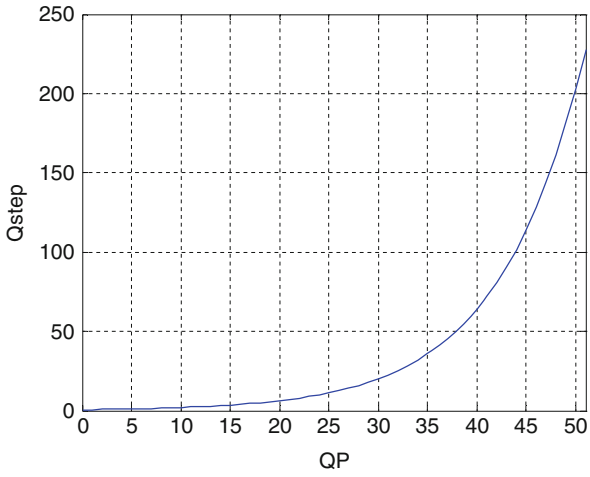


Fig. 6.6 Relationship between quantization step size (*Qstep*) and quantization parameter (*QP*)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00167

where $shift1 = (M - 5 + B)$ and $offset_{IQ} = 1 << (M - 6 + B)$. Note that the quantization matrix weights $w[x][y]$ modulate the quantization step size used for *level* at different positions in the transform block leading to a frequency-dependent quantization.

The scale factor S_{IQ} of Fig. 6.3 is equal to $2^{-shift1}$ and is obtained as follows: When $QP = 4$ (i.e., $Qstep = 1$) and there is no frequency dependent scaling (i.e., $w[x][y] = 16$), the combined scaling of the inverse transform and de-quantization in Fig. 6.3 when multiplied together should result in a product of 1 to maintain the norm of the residual block through inverse transform and inverse quantization, i.e.,

$$S_{IQ} \times g_4 \times 16 \times 2^{-(15-B-M)} = 1 \tag{6.9}$$

This results in $S_{IQ} = 2^{-(M-5+B)}$ leading to $shift1$ being equal to right shift by $(M - 5 + B)$. The scale factor $2^{-(15-B-M)}$ in (6.9) is obtained from Table 6.3.

For the output sample of the forward transform, $coeff[x][y]$, a straightforward quantization scheme can be implemented as follows:

$$\begin{aligned} &level[x][y] = sign(coeff[x][y]) \\ &* \left(\left(\left(abs(coeff[x][y]) \times f_{QP\%6} \times \frac{16}{w[x][y]} + offset_Q \right) >> \frac{QP}{6} \right) >> shift2 \right) \end{aligned} \tag{6.10}$$

where $shift2 = 29 - M - B$, and

$$\mathbf{f} = [f_0, f_1, f_2, f_3, f_4, f_5]^T = [26214, 23302, 20560, 18396, 16384, 14564]^T$$

Note that $f_{QP\%6} \approx 2^{14}/G_{QP\%6}$. The value of $shift2$ is obtained by imposing similar constraints on the combined scaling in the forward transform and the quantizater as in (6.9), i.e., $S_Q \times f_4 \times 2^{15-B-M} = 1$, where $S_Q = 2^{-shift2}$.

Finally, $offset_Q$ is chosen to achieve the desired rounding.

To summarize, the quantizer multipliers, f_i, and dequantizer multipliers, g_i, were chosen to satisfy the following conditions

- Ensure that g_i can be represented with signed 8 bit data type (i.e., $g_i < 2^7, i = 0, \ldots, 5$)
- Ensure an almost equal increase in step size from one QP value to the next (approximately 12 %) (i.e., $g_{i+1}/g_i \approx 2^{1/6},\ i = 0, \ldots, 4$ and $2g_0/g_5 \approx 2^{1/6}$)
- Ensure approximately unity gain through the quantization and de-quantization processes (i.e., $f_i \times g_i \times 16 \approx 1 << (shift1 + shift2) = 2^6 \times 2^{14} \times 16, i = 0, \ldots, 5$)
- Provide the desired absolute value of the quantization step size for $QP = 4$ (i.e. $Qstep(4) = 1$, or equivalently, $level = coeff \times 2^{-(15-B-M)}$ for $QP = 4$).

Note that the quantization equation in (6.10) is not specified in the HEVC standard and the encoder has flexibility to implement more sophisticated quantization schemes such as the rate-distortion optimized quantization (RDOQ) scheme implemented in the HEVC Test Model [13]. The idea behind RDOQ is briefly described in Chap. 9.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00168

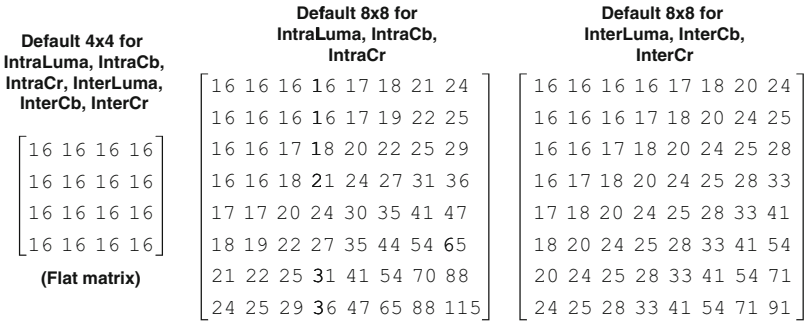


Fig. 6.7 Default quantization matrices for transform blocks of size 4×4 and 8×8

6.3.1 Quantization Matrix

In HEVC, the encoder can signal whether or not to use quantization matrices enabling frequency dependent scaling. Frequency dependent scaling is useful to carry out human visual system (HVS)-based quantization where low frequency coefficients are quantized with a finer quantization step size when compared to high frequency coefficients in the transform block [12]. HVS-based quantization can provide better visual quality than frequency independent quantization on some video sequences. HEVC uses the following 20 quantization matrices that depend on the size and type of the transform block:

- Luma: Intra 4×4, Inter 4×4, Intra 8×8, Inter 8×8, Intra 16×16, Inter 16×16, Intra 32×32, Inter 32×32
- Cb: Intra 4×4, Inter 4×4, Intra 8×8, Inter 8×8, Intra 16×16, Inter 16×16
- Cr: Intra 4×4, Inter 4×4, Intra 8×8, Inter 8×8, Intra 16×16, Inter 16×16

When frequency dependent scaling is enabled by using the syntax element `scaling_list_enabled_flag`, the quantization matrices of sizes 4×4 and 8×8 have default values as shown in Fig. 6.7. The default quantization matrices for transform blocks of size 16×16 and 32×32 are obtained from the default 8×8 quantization matrices of the same type by upsampling using replication as shown in Fig. 6.8. The red colored blocks in the figure indicate that a quantization matrix entry in the 8×8 quantization matrix is replicated into a 2×2 region in the 16×16 quantization matrix and into a 4×4 region in the 32×32 quantization matrix. 8×8 matrices are used to represent 16×16 and 32×32 quantization matrices in order to reduce the memory needed to store the quantization matrices.

Non-default quantization matrices can also be optionally transmitted in the bitstream in sequence parameter sets (SPS) or picture parameter sets (PPS). Quantization matrix entries are scanned using an up-right diagonal scan and DPCM coded and transmitted. For 16×16 and 32×32 quantization matrices, only size 8×8 matrices (which then get upsampled to the correct size in the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00169

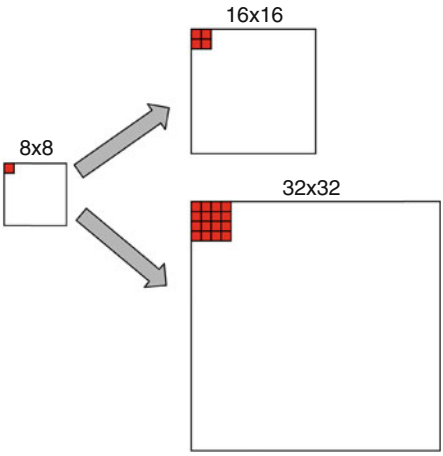


Fig. 6.8 Construction of default quantization matrices for transform block sizes 16×16 and 32×32 by using the default quantization matrix of size 8×8

Table 6.4 Quantization group size for different coding tree unit sizes

`diff_cu_qp_delta_depth`	Quantization group size for 64×64 CTU	Quantization group size for 32×32 CTU	Quantization group size for 16×16 CTU
0	64×64	32×32	16×16
1	32×32	16×16	8×8
2	16×16	8×8	–
3	8×8	–	–

decoder as shown in Fig. 6.8) and the quantization matrix entry at the DC (zero-frequency) position are transmitted. HEVC also allows for prediction of a quantization matrix from another quantization matrix of the same size. The use of quantization matrix (termed as scaling matrix in HEVC) is enabled by setting the flag `scaling_list_enabled_flag` in SPS. When this flag is enabled, additional flags in SPS and PPS control whether the default quantization matrices or non-default quantization matrices are used.

6.3.2 QP Parameter Derivation

The quantization step size (and therefore the *QP* value) may need to be changed within a picture for e.g. rate control and perceptual quantization purposes. HEVC allows for transmission of a delta *QP* value at a *quantization group* (QG) level to allow for *QP* changes within a picture. This is similar to H.264/AVC that allows for modification of QP values at a macroblock level. The QG size is a multiple of coding unit size that can vary from 8×8 to 64×64 depending on the coding tree unit (CTU) size and the syntax element `diff_cu_qp_delta_depth` as shown in Table 6.4.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00170

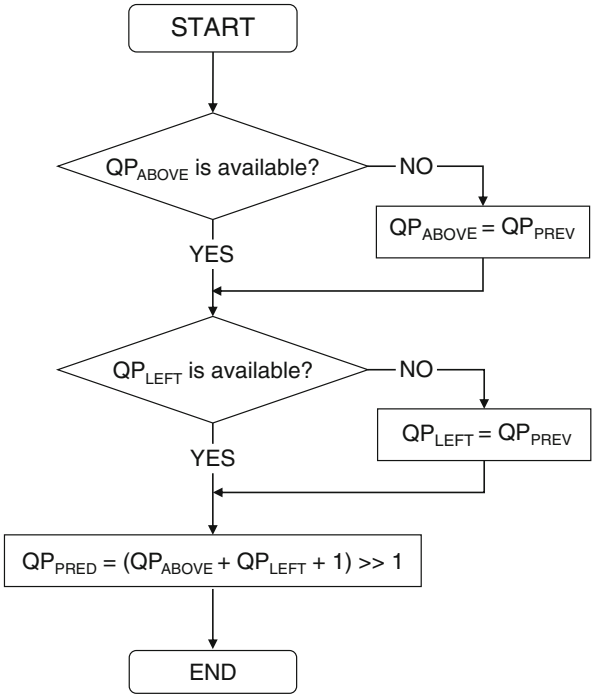


Fig. 6.9 QP predictor calculation using QP values from the left, above and previous QGs [21]

The delta *QP* is transmitted only in coding units with non-zero transform coefficients. If the CTU is split into coding units that are greater than the QG size, then delta *QP* is signaled at a coding unit (with non-zero transform coefficients) that is greater than the QG size. If the CTU is split into coding units that are smaller than the QG size, then the delta *QP* is signaled in the first coding unit with non-zero transform coefficients in the QG. If a QG has coding units with all zero transform coefficients (e.g. if the merge mode is used in all the coding units of the QG), then delta *QP* will not be signaled.

The QP predictor used for calculating the delta QP uses a combination of QP values from the left, above and the previous QG in decoding order as shown in Fig. 6.9 [21]. The QP predictor uses a combination of two predictive techniques: spatial QP prediction (from left and above QGs) and previous QP prediction. It uses spatial prediction from left and above within a CTU and uses the previous QP as predictor at the CTU boundary. This is shown in Fig. 6.9. The spatially adjacent QP values, QP_{LEFT} and QP_{ABOVE} are considered to be not available when they are in a different CTU or if the current QG is at a slice/tile/picture boundary. When a spatially adjacent QP value is not available, it is replaced with the previous QP value, QP_{PREV}, in decoding order. The previous QP, QP_{PREV}, is initialized to the slice QP value at the beginning of the slice, tile or wavefront.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00171

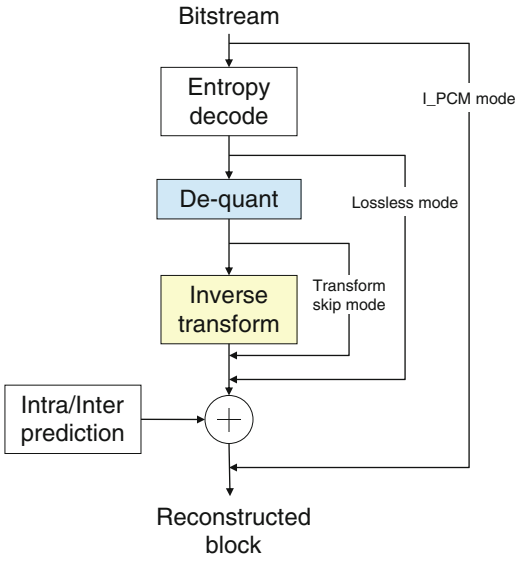


Fig. 6.10 I_PCM, lossless and transform skip modes in decoder

The QP derivation process described in this subsection is used for calculating the luma QP value. The chroma QP values (one for the Cr component and one for the Cb component) are derived from the luma QP by using picture level and slice level offsets and a table lookup.

6.4 HEVC Special Coding Modes

HEVC has three special modes that modify the transform and quantization process: (a) I_PCM mode [8], (b) lossless mode [31], and (c) transform skip mode [19]. These modes skip either the transform or both the transform and quantization. Figure 6.10 shows these modes on top of the generic video decoder data flow of Fig. 6.1.

- In the I_PCM mode, both transform and transform-domain quantization are skipped. In addition, entropy coding and prediction are skipped too and the video samples are directly coded with the specified PCM bit depth. The I_PCM mode is designed for use when there is data expansion during coding e.g. when random noise is input to the video codec. By directly coding the video samples, the data expansion can be avoided for such extreme video sequences. The IPCM mode is signaled at the coding unit level using the syntax element `pcm_flag`.
- In the lossless mode, both transform and quantization are skipped. (The in-loop filter which is not shown in Fig. 6.1 is skipped too.) Mathematically lossless reconstruction is possible since the residual from inter- or intra-picture prediction is directly coded. The lossless mode is signaled at a coding unit level (using the syntax element `cu_transquant_bypass_flag`) in order to enable

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00172

mixed lossy/lossless coding of pictures. Such a feature is useful in coding video sequences with mixed content, e.g. natural video with overlaid text and graphics. The text and graphics regions can be coded losslessly to maximize readability whereas the natural content can be coded in a lossy fashion.

- In the transform skip mode, only the transform is skipped. This mode was found to improve compression of screen-content video sequences generated in applications such as remote desktop, slideshows etc. These video sequences predominantly contain text and graphics. Transform skip is restricted to only 4×4 transform blocks and its use is signaled at the transform unit level by the `transform_skip_flag` syntax element.

6.5 Complexity Analysis

With straightforward matrix multiplication, the number of operations for the 1D inverse transform is N^2 multiplications and $N(N-1)$ additions. For the 2D transform, the number of multiplications required is $2N^3$ and the number of additions required is $2N^2(N-1)$. However, by utilizing the (anti-) symmetry properties of each basis vector inherited from DCT, the number of arithmetic operations can be significantly reduced. We refer to the algorithm that does this as the Even–Odd decomposition in this paper (it was also referred to as partial butterfly during HEVC development) [14]. Even–Odd decomposition is illustrated below using the 4- and 8-point inverse transform.

Consider the 4-point forward transform matrix defined in (6.5). For notational simplicity the constants $d_{i,0}^{32}$ of Eq. (6.5) will be denoted by d_i. Using the new notation (6.5) becomes

$$\mathbf{D}_4 = \begin{bmatrix} d_{16} & d_{16} & d_{16} & d_{16} \\ d_8 & d_{24} & -d_{24} & -d_8 \\ d_{16} & -d_{16} & -d_{16} & d_{16} \\ d_{24} & -d_8 & d_8 & -d_{24} \end{bmatrix} \tag{6.11}$$

The inverse transform matrix is given by $\mathbf{D}_4^T$. Let $\mathbf{x} = [x_0, x_1, x_2, x_3]^T$ be the input vector and $\mathbf{y} = [y_0, y_1, y_2, y_3]^T$ denote the output. The 1D 4-point inverse transform is given by the following equation:

$$\mathbf{y} = \mathbf{D}_4^T \mathbf{x} \tag{6.12}$$

The Even–Odd decomposition of the inverse transform of an N-point input consists of the following three steps:

1. Calculate the even part using a $N/2 \times N/2$ subset matrix obtained from the even columns of the inverse transform matrix (6.13 shows an example).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00173

2. Calculate the odd part using a $N/2 \times N/2$ subset matrix obtained from the odd columns of the inverse transform matrix (6.15 shows an example).
3. Add/subtract the odd and even parts to generate N-point output (6.16 shows an example).

Even–odd decomposition of the inverse 4-point transform is given by (6.14–6.16):

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \begin{bmatrix} d_{16} & d_{16} \\ d_{16} & -d_{16} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} \tag{6.13}$$

The even part can be further simplified as:

$$\begin{aligned} t_0 &= d_{16} x_0 \\ t_1 &= d_{16} x_2 \\ \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} &= \begin{bmatrix} t_0 + t_1 \\ t_0 - t_1 \end{bmatrix} \end{aligned} \tag{6.14}$$

Odd part:

$$\begin{bmatrix} z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -d_{24} & d_8 \\ -d_8 & -d_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{6.15}$$

Add/sub:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} z_0 - z_3 \\ z_1 - z_2 \\ z_1 + z_2 \\ z_0 + z_3 \end{bmatrix} \tag{6.16}$$

The direct 1D 4-point transform using (6.12) would require 16 multiplications and 12 additions. The 2D transform will require 128 multiplications and 96 additions. Even–Odd decomposition on the other hand requires a total of six multiplications and eight additions for 1D transform using (6.14–6.16). The 2D transform using Even–Odd decomposition will require a total of 48 multiplications and 64 additions which is 62.5 % savings in number of multiplications and 33.3 % savings in number of additions when compared to direct matrix multiplication.

The 8-point 1D inverse transform is defined by the following equation:

$$\mathbf{y} = \mathbf{D}_8^T \mathbf{x} \tag{6.17}$$

where $\mathbf{x} = [x_0, x_1, \ldots, x_7]^T$ is input and $\mathbf{y} = [y_0, y_1, \ldots, y_7]^T$ is output, and $\mathbf{D}_8$ is given by:

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00174

$$\mathbf{D}_8 = \begin{bmatrix} d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} \\ d_4 & d_{12} & d_{20} & d_{28} & -d_{28} & -d_{20} & -d_{12} & -d_4 \\ d_8 & d_{24} & -d_{24} & -d_8 & -d_8 & -d_{24} & d_{24} & d_8 \\ d_{12} & -d_{28} & -d_4 & -d_{20} & d_{20} & d_4 & d_{28} & -d_{12} \\ d_{16} & -d_{16} & -d_{16} & d_{16} & d_{16} & -d_{16} & -d_{16} & d_{16} \\ d_{20} & -d_4 & d_{28} & d_{12} & -d_{12} & -d_{28} & d_4 & -d_{20} \\ d_{24} & -d_8 & d_8 & -d_{24} & -d_{24} & d_8 & -d_8 & d_{24} \\ d_{28} & -d_{20} & d_{12} & -d_4 & d_4 & -d_{12} & d_{20} & -d_{28} \end{bmatrix} \tag{6.18}$$

Even–Odd decomposition for the 8-point inverse transform is given by (6.19–6.21).

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} d_{16} & d_8 & d_{16} & d_{24} \\ d_{16} & d_{24} & -d_{16} & -d_8 \\ d_{16} & -d_{24} & -d_{16} & d_8 \\ d_{16} & -d_8 & d_{16} & -d_{24} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} \tag{6.19}$$

Odd part:

$$\begin{bmatrix} z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} -d_{28} & d_{20} & -d_{12} & d_4 \\ -d_{20} & d_4 & -d_{28} & -d_{12} \\ -d_{12} & d_{28} & d_4 & d_{20} \\ -d_4 & -d_{12} & -d_{20} & -d_{28} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix} \tag{6.20}$$

Add/sub:

$$\mathbf{y} = [z_0 - z_7, z_1 - z_6, z_2 - z_5, z_3 - z_4, z_3 + z_4, z_2 + z_5, z_1 + z_6, z_0 + z_7]^T \tag{6.21}$$

Note that the even part of the 8-point inverse transform is actually a 4-point inverse transform (by comparing 6.19 with transpose of $\mathbf{D}_4$ in 6.11) i.e.,

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \mathbf{D}_4^T \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} \tag{6.22}$$

So the Even–Odd decomposition of the 4-point inverse transform (6.14–6.16) can be used to further reduce computational complexity of the even part of the 8-point transform in (6.19).

The direct 1D 8-point transform using (6.17) would require 64 multiplications and 56 additions. The 2D transform will require 1,024 multiplications and 896 additions. An even–odd decomposition on the other hand requires 6 multiplications for (6.22) and 16 multiplications for (6.20) resulting in a total of 22 multiplications. It requires 8 additions for (6.22), 12 additions for (6.20) and 8 additions for

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00175

(6.21) resulting in a total of 28 additions. The 2D transform using Even–Odd decomposition will require a total of 352 multiplications and 448 additions.

The computational complexity calculation for the 4-point and 8-point inverse transform can be extended to inverse transforms of larger sizes. In general, the resulting number of multiplications and additions (excluding the rounding operations associated with the shift operations) for the two-dimensional N-point inverse transform can be shown to be

$$O_{mult} = 2N\left(1 + \sum_{k=1}^{\log_2 N} 2^{2k-2}\right)$$

$$O_{add} = 2N\left(\sum_{k=1}^{\log_2 N} 2^{k-1}\left(2^{k-1}+1\right)\right)$$

The number of arithmetic operations for the inverse transform can be further reduced if knowledge about zero-valued input transform coefficients is assumed. In an HEVC decoder, this information can be obtained from the entropy decoding or de-quantization process. For typical video content many blocks of size $N \times N$ will have non-zero coefficients only in a $K \times K$ low frequency sub-block. For example in [5] it was found that on average around 75 % of the transform blocks had non-zero coefficients only in $K \times K$ low frequency sub-blocks. Computations can be saved in two ways for such transform blocks. Figure 6.11 shows the first way. Columns that are completely zero need not be inverse transformed. So only K 1D IDCTs along columns needs to be carried out. However, all N rows will need to be transformed subsequently. The second way to reduce computations is by exploiting the fact that each of the column and row IDCT is on a vector that has non-zero values only in the first K locations. For example with $K = N/2$, $x_4 = x_5 = x_6 = x_7 = 0$, roughly half the computations for the inverse transformation can be eliminated by simplifying Eqs. (6.19–6.20) to

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} d_{16} & d_8 \\ d_{16} & d_{24} \\ d_{16} & -d_{24} \\ d_{16} & -d_8 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix}$$

Odd part:

$$\begin{bmatrix} z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} -d_{28} & d_{20} \\ -d_{20} & d_4 \\ -d_{12} & d_{28} \\ -d_4 & -d_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00176

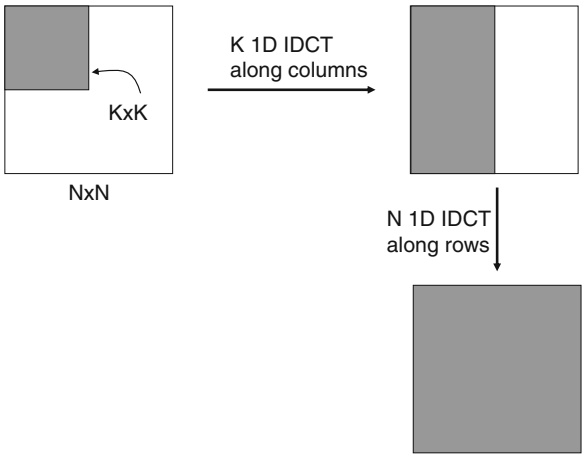


Fig. 6.11 Efficient implementation of inverse transform of a block with non-zero coefficients in only the $K \times K$ low frequency sub-block. Shaded regions denote the regions that can contain non-zero coefficients. Only K 1D IDCTs are required along columns

Table 6.5 Arithmetic operation counts for HEVC two-dimensional inverse transforms

N	K	O_{mult}	O_{add}
4	4	48	64
8	8	352	448
8	4	132	228
16	16	2,752	3,200
16	8	1,032	1,512
16	4	420	820
32	32	21,888	23,808
32	16	8,208	10,320
32	8	3,400	5,320
32	4	1,476	3,060

In general, the number of multiplications can be reduced approximately by a factor of $(N/K)^2$ for the first stage and a factor of (N/K) for the second stage. Table 6.5 shows the number of arithmetic operations for various values of N and K.

Note that the majority of the arithmetic operations listed in Table 6.5 can be efficiently implemented using SIMD instructions since the operations are matrix multiply operations. For example, for an 8×8 inverse transform implementation, (6.20) can be efficiently implemented on a 4-way SIMD processor in 4 cycles v/s 16 cycles on a processor without SIMD acceleration. Software performance using SIMD acceleration on various Intel processor architectures for the 8×8, 16×16, and 32×32 transform sizes are provided in [3, 11].

Only the Even–Odd decomposition of the inverse transform has been described in this subsection. However, the Even–Odd decomposition idea can be used to reduce the complexity of the forward transform too. The article [6] presents analysis of both the forward and the inverse core transform in more details. It also describes hardware sharing by the application of property 4 of Sect. 6.2.1 (smaller transforms being embedded in larger transforms).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00177

6.6 Coding Performance

The different transform sizes used in a coding block in HEVC are signaled in a quadtree structure [29]. The maximum transform size to use in a coding block is signaled in the sequence parameter set. Table 6.6 compares the coding performance of HEVC when all transform sizes (up to 32×32) are used to the coding performance when only 4×4 and 8×8 transforms are used as in H.264/AVC. The standard Bjøntegaard Delta-Rate (BD-Rate) metric [2] is used for comparison. Table 6.6 shows that there is a bit rate savings in the range of 5.6–6.8 % on average because of the introduction of larger transform sizes (16×16 and 32×32) in HEVC. The bit rate savings are higher at larger resolution video such as 4K (2560×1600) and 1080p (1920×1080). The HEVC Test Model, HM-9.0.1 [13] was used for the simulations and the video sequences and coding conditions used were as described in [4].

Table 6.7 compares the coding performance of the HEVC 4×4 and 8×8 transforms to that of the corresponding H.264/AVC transforms. The H.264/AVC 4×4 and 8×8 transforms were converted to 8-bit precision and implemented in the HM-9.0.1 Test Model. Only the 4×4 and 8×8 transform sizes were enabled in the simulations. It can be seen from Table 6.7 that the HEVC 4×4 and 8×8 transforms perform better than the corresponding H.264/AVC transforms in terms of coding performance.

Table 6.6 BD-rate savings of using larger transform sizes (16×16 and 32×32) on top of the smaller transform sizes (4×4 and 8×8)

	All Intra (%)	Random access (%)	Low delay B (%)
4K	−9.1	−10.1	n/a
1080p	−6.7	−8.0	−9.1
WVGA	−2.5	−4.3	−6.0
WQVA	−2.2	−2.8	−3.7
720p	−7.7	n/a	−8.4
Overall	−5.6	−6.4	−6.8

Table 6.7 BD-Rate savings of the HEVC 4×4 and 8×8 transforms versus the H.264/AVC 4×4 and 8×8 transforms

	All Intra (%)	Random access (%)	Low delay B (%)
4K	−1.2	−0.7	n/a
1080p	−0.6	−0.4	−0.3
WVGA	−0.2	−0.2	−0.1
WQVA	−0.1	0.0	−0.1
720p	−0.5	n/a	−0.2
Overall	−0.5	−0.3	−0.2

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00178

References

1. Alshina E, Alshin A, Lee W, Park J (2011) Full factorization core transforms for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G737, Geneva, Nov. 2011.
2. Bjøntegaard G (2001) VCEG-M33: calculation of average PSNR differences between RD curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001.
3. Bossen F (2011) On software complexity, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G757, Geneva, Nov. 2011.
4. Bossen F (2012) Common HM test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-K1100, Shanghai, Oct. 2012.
5. Budagavi M (2011) IDCT pruning, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E386, Geneva, Mar. 2011
6. Budagavi M, Fuldseth A, Bjøntegaard G, Sze V, Sadafale M (2013) Core transform design in the High Efficiency Video Coding (HEVC) standard. IEEE Trans Circuits Syst Video Technol 7(6):1029–1041
7. Chen W-H, Smith CH, Fralick S (1977) A fast computational algorithm for the discrete cosine transform. IEEE Trans Commun COM-25(9):1004–1009
8. Chono K, Aoki H, Wahadaniah V, Lim CS (2011) Proposal of enhanced PCM coding in HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E192, Geneva, Mar. 2011
9. Dai W, Krishnan M, Topiwala J, Topiwala P, Alshina E (2011) Lossless core transforms for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G266, Geneva, Nov. 2011
10. Fuldseth A, Bjøntegaard G, Sadafale M, Budagavi M (2011) Core transform design for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G495, Geneva, Nov. 2011
11. Fuldseth A, Endresen LP, Selnes S, Arbatov V, Franchetti F, Puschel M (2011) SIMD Optimization of proposed HEVC transforms, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G497, Geneva, Nov. 2011
12. Haque M, Tabatabai A, Morigami Y (2011) HVS model based default quantization matrices, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G880, Geneva, Nov. 2011
13. HEVC Test Model HM-9.0.1 Nov. 2012 [Online]. Available https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.0.1/
14. Hung C-Y, Landman P (1997) Compact inverse discrete cosine transform circuit for MPEG video decoding. In: Proceedings of IEEE SIPS, Nov. 1997, pp 364–373
15. ITU-T Rec. H.264 and ISO/IEC 14496-10 (2003) Advanced video coding
16. ITU-T Rec. H.265 and ISO/IEC 23008-2 (2013) High efficiency video coding
17. Joshi R, Sole J, Karczewicz M (2011) Scaled integer transform supporting recursive factorization structure, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G579, Geneva, Nov. 2011
18. Kerofsky L, Riabtsev S (2012) Dynamic range analysis of HEVC/H.265 inverse transform operations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L0332, Geneva, Jan. 2013
19. Lan C, Xu J, Sullivan GJ, Wu F (2012) Intra transform skipping, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0408, Geneva, Apr. 2012
20. Malvar HS, Hallapuro A, Karczewicz M, Kerofsky L (2003) Low complexity transform and quantization in H.264/AVC. IEEE Trans Circuits Syst Video Technol 13(7):598–603
21. Nakamura H, Nishitani M, Fukushima S (2012) Non-CE4: compatible QP prediction with RC and AQ, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0204, San Jose, Feb. 2012

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00179

22. Rao KR, Yip P (1990) Discrete cosine transform: algorithms, advantages, applications. Academic, Boston
23. Sadafale M, Budagavi M (2010) Low-complexity configurable transform architecture for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C226, Guangzhou, Oct. 2010
24. Saxena A, Fernandes FC (2011) CE7: mode-dependent DCT/DST without 4 × 4 full matrix multiplication for intra prediction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E125, Geneva, Mar. 2011
25. Saxena A, Fernandes FC (2013) DCT/DST-based transform coding for intra prediction in image/video coding. IEEE Trans Image Proc 22(10):3974–3981
26. Tikekar M, Huang C-T, Juvekar C, Chandrakasan A (2011) Core transform property for practical throughput hardware design. Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G265, Geneva, Nov. 2011
27. Wiegand T, Sullivan GJ, Bjøntegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–570
28. Wiegand T, Han W-J, Ohm J-R, Sullivan GJ (2010) High Efficiency Video Coding (HEVC) text specification working draft 1, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C403, Guangzhou, Oct. 2010
29. Winken M, Helle P, Marpe D, Schwarz H, Wiegand T (2011) Transform coding in the HEVC Test Model. In: Proceedings of the IEEE international conference image processing, pp 3693–3696
30. Zhou M, Sze V (2010) TE 12: evaluation of transform unit (TU) size, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C056, Guangzhou, Oct. 2010
31. Zhou M, Gao W, Jiang M, Yu H (2012) HEVC lossless coding and improvements. IEEE Trans Circuits Syst Video Technol 22(12):1839–1843

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00180

Chapter 7
In-Loop Filters in HEVC

Andrey Norkin, Chih-Ming Fu, Yu-Wen Huang, and Shawmin Lei

Abstract The HEVC standard specifies two in-loop filters, a deblocking filter and a sample adaptive offset (SAO). The in-loop filters are applied in the encoding and decoding loops, after the inverse quantization and before saving the picture in the decoded picture buffer. The deblocking filter is applied first. It attenuates discontinuities at the prediction and transform block boundaries. The second in-loop filter, SAO, is applied to the output of the deblocking filter and further improves the quality of the decoded picture by attenuating ringing artifacts and changes in sample intensity of some areas of a picture. The most important advantage of the in-loop filters is improved subjective quality of reconstructed pictures. In addition, using the filters in the decoding loop also increases the quality of the reference pictures and hence also the compression efficiency.

7.1 Introduction

The in-loop filters constitute an important part of the HEVC video coding standard. As seen from the name, the in-loop filters are applied in the encoding and decoding loops, after the inverse quantization but before saving the picture to the decoded picture buffer. HEVC standard specifies two in-loop filters, a deblocking filter, which is applied first, and a sample adaptive offset (SAO), which is applied to the output of the [40] deblocking filter. The deblocking filter attenuates discontinuities at prediction and transform block boundaries [36, 37]. The SAO further improves

A. Norkin (✉)
Ericsson Research, Ericsson, Stockholm, Sweden
e-mail: andrey.norkin@ericsson.com

C.-M. Fu • Y.-W. Huang • S. Lei
MediaTek, Hsinchu, Taiwan
e-mail: chihming.fu@mediatek.com; yuwen.huang@mediatek.com; shawmin.lei@mediatek.com

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_7

© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00181

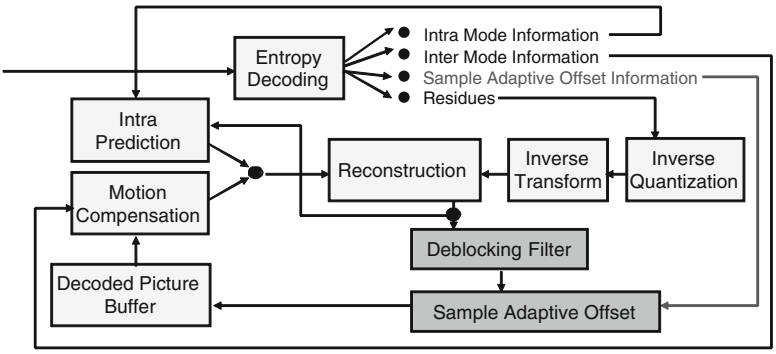


Fig. 7.1 Block diagram of HEVC decoder. Reproduced with permission from [13], © 2012 IEEE

the quality of the decoded picture by reducing the ringing artifacts and changes in the sample intensity of areas of a reconstructed picture. Since the deblocking and SAO attenuate different artifacts, their benefits are additive when used together. An HEVC encoder can turn on and off each of the in-loop filters independently.

Modern video coding standards try to remove as much redundancy from the coded representation of video as possible. One of the sources of redundancy is the temporal redundancy, i.e. similarity between the subsequent pictures in a video sequence. This type of redundancy is effectively removed by motion prediction. Another type of redundancy is spatial redundancy and is removed by intra-prediction from the neighboring samples and spatial transforms. In HEVC, both the motion prediction and transform coding are block-based. The size of motion-predicted blocks varies from 8×4 and 4×8, to 64×64 luma samples, while the size of block transforms and intra-predicted blocks varies from 4×4 to 32×32 samples.

These blocks are coded relatively independently from the neighboring blocks and approximate the original signal with some degree of similarity. Since coded blocks only approximate the original signal, the difference between the approximations may cause discontinuities at the prediction and transform block boundaries [27, 36, 37]. These discontinuities are attenuated by the deblocking filter.

Larger transforms used in HEVC can also introduce more ringing artifacts compared to H.264/AVC that mainly come from quantization error of transform coefficients [17]. In addition to that, HEVC uses 8 or 7-tap fractional luma sample interpolation and 4-tap fractional chroma sample interpolation, while H.264/AVC uses 6-tap and 2-tap for luma and chroma respectively. A higher number of interpolation taps can also lead to more ringing artifacts. These artifacts are corrected by a new filter: sample adaptive offset (SAO). As shown in Fig. 7.1, SAO is applied to the output of the deblocking filter when the deblocking filter is turned on, otherwise, it is applied to the reconstructed picture

There are several reasons for making in-loop filters a part of the standard. In principle, the in-loop filters can also be applied as post-filters. An advantage of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00182

using post-filters is that decoder manufacturers can create post-filters that better suit their needs. However, if the filter is a part of the standard, the encoder has control over the filter and can assure the necessary level of quality by instructing the decoder to enable the filter and specifying the filter parameters. Moreover, since the in-loop filters increase the quality of the reference pictures, they also improve the compression efficiency. A post-filter would also require an additional buffer for filtered pictures, while the output of an in-loop filter can be kept in the decoded picture buffer (DPB). There is also another specific advantage of using the deblocking filter as an in-loop filter compared to the deblocking post-filter. If the deblocking is applied as a post-filter, block artifacts can be copied by motion estimation inside the blocks, which can make the artifacts detection more difficult and increases the deblocking complexity compared to the in-loop filtering, which needs to be applied only to the block boundaries [27, 36, 37].

It is known that the deblocking in-loop filter in H.264/AVC constitutes a significant part of the decoder complexity [27]. Therefore, when designing the in-loop filters in HEVC, efforts have been spent on reducing the loop filters complexity, while still achieving improvements of subjective quality. The HEVC in-loop filters are easily parallelizable, which can bring advantages when running the HEVC decoders and encoders on multi-core architectures.

This chapter is organized as follows. Section 7.2 of the chapter describes the HEVC deblocking filter. Section 7.3 describes SAO, in particular, two types of sample offsets: an edge offset and a band offset. Section 7.4 discusses implementation and parallelization aspects of the HEVC in-loop filters as well as the details of CTU-based implementation of the filter operations. Section 7.5 demonstrates the subjective quality and coding efficiency improvements. Section 7.6 summarizes the main differences between the in-loop filters in HEVC and H.264/AVC while Sect. 7.7 concludes the chapter.

7.2 Deblocking Filter

7.2.1 Block Artifacts in Video Coding

As mentioned in Sect. 7.1, in HEVC both the motion prediction and transform coding are block-based. The size of motion predicted blocks varies from 4×8 and 8×4 to 64×64 luma samples, while the size of block transforms and intra-predicted blocks varies from 4×4 to 32×32 samples. These blocks are coded relatively independently from their neighboring blocks and approximate the original signal with some degree of similarity. Since coded blocks only approximate the original signal, the difference between the approximations may cause discontinuities at the prediction and transform block boundaries. For example, motion prediction of the adjacent blocks may come from the non-adjacent areas of a reference picture (see Fig. 7.2) or even from different reference pictures. In case of non-overlapping block transforms,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00183

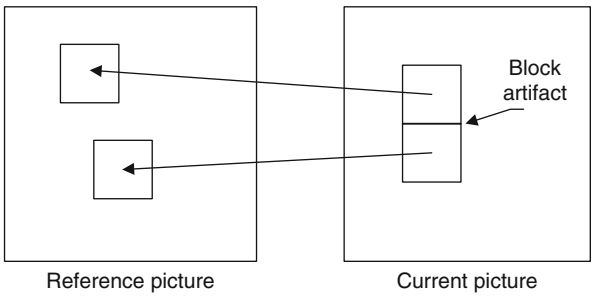


Fig. 7.2 Block artifact may be created when adjacent blocks are predicted from non-adjacent areas in the reference picture

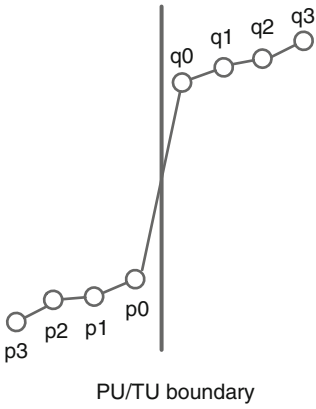


Fig. 7.3 Example of block artifact in one dimension [37]

used in HEVC, coarse quantization can also create discontinuities at the block boundaries. In highly detailed areas with high-frequency content, such artifacts can be masked by the human visual system. However, in the smooth areas, discontinuities between the blocks are easily noticed by a viewer and may cause significant degradation of the perceived video quality. The example of a block artifact in one dimension is shown in Fig. 7.3. The horizontal axis shows the sample positions along a horizontal or vertical 1-D line, and the vertical axis shows the sample values.

Deblocking filter attenuates the artifacts in the areas, where they are mostly visible, i.e. in the smooth, uniform areas. The excessive filtering in the highly detailed areas should be avoided since it can cause undesirable blurring. The artifacts in those areas are rarely noticed by the human eye, while it is also more difficult to determine whether the discontinuity is caused by a block boundary or belongs to the original signal [27]. Therefore, an important part of the deblocking filter is the deblocking filtering decisions, which determine whether a particular part of a block boundary is to be filtered. In these decisions, the HEVC deblocking filter uses the mode and motion information from the decoded bitstream as well as analyses the values of reconstructed samples on the sides of the block boundary. The strength of the deblocking filter can also be adjusted by the encoder on the picture and the slice basis.

Section 7.2.2 provides a description of the HEVC deblocking filer, while Sect. 7.5.1 discusses the coding efficiency and subjective quality improvements

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00184

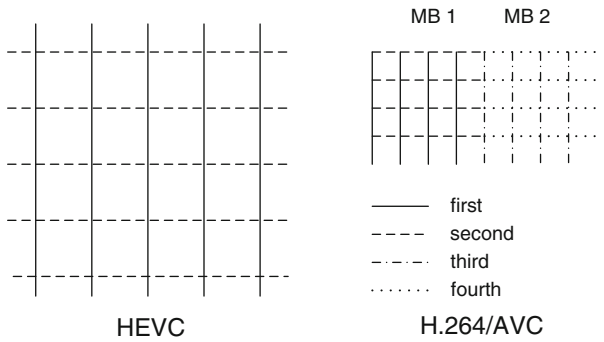


Fig. 7.4 Order of boundaries processing in HEVC and H.264/AVC deblocking. In each group (first to fourth), boundaries are processed from left to right and from top to bottom. In HEVC all vertical (horizontal) boundaries can be processed in parallel

brought by HEVC deblocking filtering. The deblocking filter complexity and parallelization aspects are addressed in Sect. 7.4.1.

7.2.2 HEVC Deblocking Filter Description

7.2.2.1 Decisions to Filter a Block Boundary

As a compromise between the subjective quality and computational complexity, the HEVC deblocking filter in case of 4:2:0 chroma subsampling is only applied to the block boundaries that lie at the luma and chroma sample positions that are multiples of eight (in H.264/AVC the deblocking is applied on the 4×4 luma and chroma sample grid). Since the deblocking filtering is only applied to the boundaries between the coding units (CU), prediction units (PU), or transform units (TU) and not to the inside areas, the average complexity of the HEVC deblocking is further decreased compared to the H.264/AVC since HEVC can use larger block sizes.

In HEVC deblocking, the vertical boundaries in a picture are filtered first, followed by the horizontal boundaries. In a coding unit, the vertical boundaries between the coding blocks are processed starting from the left-most boundary towards the right-hand side. The horizontal boundaries are processed starting from the top-most boundary towards the bottom. In contrast to that, the H.264/AVC deblocking filter operates on a macroblock (MB) basis, where the four vertical boundaries in an MB are processed sequentially starting from the left-most MB boundary, and then the four horizontal MB boundaries are processed starting from the top-most MB boundary. The order of processing of the block boundaries in HEVC and H.264/AVC is compared in Fig. 7.4. One can see that the filtering order in HEVC deblocking is more regular than that in H.264/AVC. Moreover, since the deblocking of one vertical (horizontal) boundary in HEVC does not affect deblocking of other vertical (horizontal) boundaries because of only filtering the boundaries on the 8×8 sample grid, all the vertical (horizontal) boundaries can be

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00185

Table 7.1 Boundary strength (*Bs*) derivation [37]

Conditions	Bs
At least one of the adjacent blocks is intra	2
At least one of the adjacent blocks has non-zero transform coefficients	1
Absolute difference between the motion vectors that belong to the adjacent blocks is greater than or equal to one integer luma sample	1
Motion prediction in the adjacent blocks refers to different reference pictures or number of motion vectors is different	1
Otherwise	0

processed in parallel. This allows better parallelization which is addressed in more detail in Sect. 7.4.1. The parallelization of H.264/AVC deblocking filtering is more complicated since filtering of a block boundary affects the deblocking decisions at a parallel block boundary, and the vertical and horizontal filtering operations are alternating.

A decision whether to filter a block boundary uses the bitstream information such as prediction modes and motion vectors. Some coding conditions are more likely to create strong block artifacts, which are represented by a so-called *boundary strength* (*Bs*) variable that is assigned to every block boundary and is determined as in Table 7.1. The deblocking is only applied to the block boundaries with *Bs* greater than zero for a luma component and *Bs* greater than 1 for chroma components. Higher values of *Bs* enable stronger filtering by using higher clipping parameter values. The *Bs* derivation conditions reflect the probability that the strongest block artifacts appear at the intra-predicted block boundaries. The conditions also enable luma deblocking when there is possibility of block artifacts caused by quantization and by prediction from non-adjacent areas in a reference picture. Not filtering block boundaries with *Bs* equal to zero avoids multiple repetitive filtering of static areas where the samples are just copied from one picture to another. In chroma deblocking, only the block boundaries adjacent to intra-predicted blocks are filtered, which reduces the deblocking complexity while still removing the strongest block artifacts. The algorithm for *Bs* derivation is explained in a flowchart in Fig. 7.5.

For luma block boundaries with non-zero *Bs* values, the signal on the sides of the block boundary is evaluated to decide whether the deblocking should be applied. For chroma block boundaries, no further evaluation is performed.

The deblocking decisions are made separately for each four-sample segment of a block boundary (see Fig. 7.6). Since the deblocking needs to attenuate visible artifacts in smooth areas, the deblocking decisions evaluate whether the signal at the sides of the block boundary is smooth, i.e. if the signal is flat or has a shape of an inclined plane (ramp) [36, 37, 43]. The deblocking filtering is applied to a block boundary if the following expression is evaluated to be true:

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00186

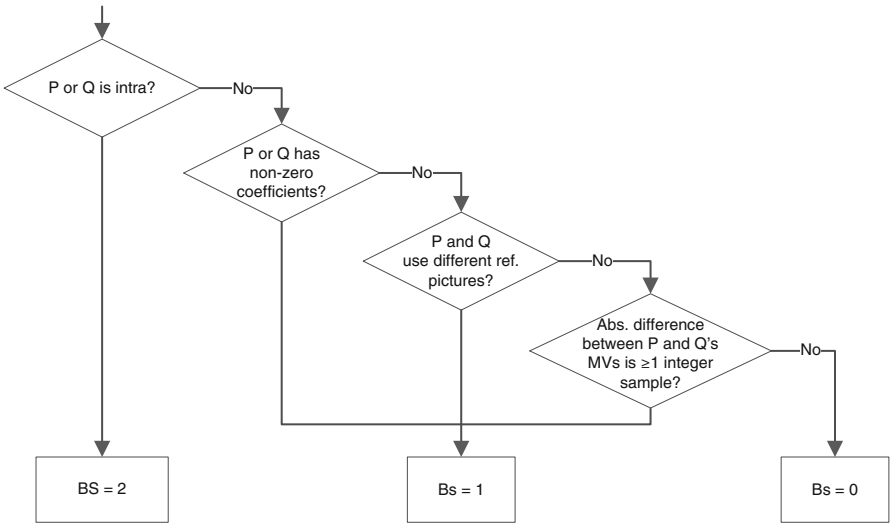


Fig. 7.5 Derivation of boundary strength (*Bs*) for block boundary. P denotes the left (or top) block and Q denotes the right (or bottom) block at the block boundary

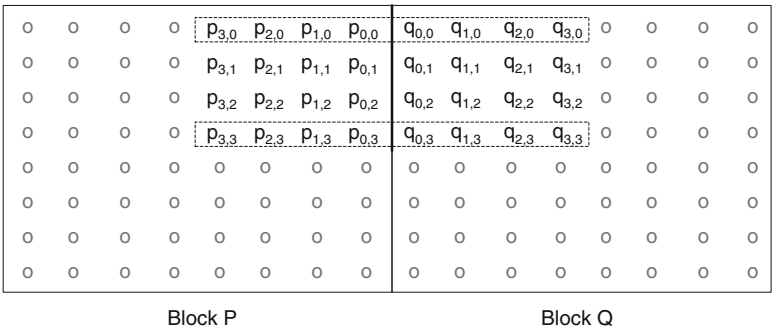


Fig. 7.6 Four-sample segment of vertical block boundary between adjacent blocks P and Q. Deblocking decision are made based on lines 0 and 3 (*dashed*)

$$|p_{2,0} - 2p_{1,0} + p_{0,0}| + |p_{2,3} - 2p_{1,3} + p_{0,3}| + |q_{2,0} - 2q_{1,0} + q_{0,0}| + |q_{2,3} - 2q_{1,3} + q_{0,3}| < \beta, \quad (7.1)$$

where the threshold β depends on the quantization parameters QP and is derived from a look-up table. Equation (7.1) is used to check how much the signal on each side of the block boundary deviates from a straight line (ramp). The equations in this book chapter are given for the case of a vertical block boundary as in Fig. 7.6. The equations for a horizontal block boundary may be obtained from the equations for the vertical boundary by swapping the row and column indices.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00187

The HEVC deblocking has two filtering modes: a normal filtering mode and a strong filtering mode. A decision between these two modes is done for each four-sample segment of a block boundary. The strong filtering is applied to the block boundary if all of the following conditions are true for lines $i = 0$ and $i = 3$ (see Fig. 7.6) [32, 37, 43].

$$|p_{2,\mathrm{i}} - 2p_{1,\mathrm{i}} + p_{0,\mathrm{i}}| + |q_{2,\mathrm{i}} - 2q_{1,\mathrm{i}} + q_{0,\mathrm{i}}| < \beta/8, \tag{7.2}$$

$$|p_{3,\mathrm{i}} - p_{0,\mathrm{i}}| + |q_{0,\mathrm{i}} - q_{3,\mathrm{i}}| < \beta/8, \tag{7.3}$$

$$|p_{0,\mathrm{i}} - q_{0,\mathrm{i}}| < 2.5t_C. \tag{7.4}$$

If all of the (7.2)–(7.4) are true, the strong filtering is applied, otherwise, the normal deblocking filter is applied. The threshold parameter t_C is the clipping parameter described later in this section. Equation (7.4) makes sure that the step between the sample values at the sides of the block boundary is small, while (7.2) checks that there are no significant signal variations at the sides of the boundary, and (7.3) verifies that the signal on both sides is flat.

The deblocking filtering decisions for a block boundary including the decisions between the strong and the normal filtering are summarized in a flowchart in Fig. 7.7.

7.2.2.2 Normal Filtering Mode

When a normal deblocking filtering mode is used, the following conditions are evaluated to decide how many samples are modified at each side of the block boundary. Condition in (7.5) determines how many samples from the block boundary are modified in block P, while condition in (7.6) determines how many samples are modified in block Q (see Fig. 7.6) [36]. The decisions use the same principle as decision (7.1). The smoother the signal on the side of the block boundary, the more filtering is applied [35].

$$|p_{2,0} - 2p_{1,0} + p_{0,0}| + |p_{2,3} - 2p_{1,3} + p_{0,3}| < 3/16\,\beta, \tag{7.5}$$

$$|q_{2,0} - 2q_{1,0} + q_{0,0}| + |q_{2,3} - 2q_{1,3} + q_{0,3}| < 3/16\ \beta. \tag{7.6}$$

If (7.5) is true, two samples from the block boundary are modified in block P, otherwise, one sample is modified. If (7.6) is true, two samples from the block boundary are modified in block Q, otherwise, one sample is modified. The decisions are made for each side of the block boundary independently, i.e. one sample may be filtered on one side of the block boundary, and two samples on the other side.

When condition in (7.1) is true for a four-sample segment of the block boundary, the deblocking filtering operations are subsequently applied to each of the four lines

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00188

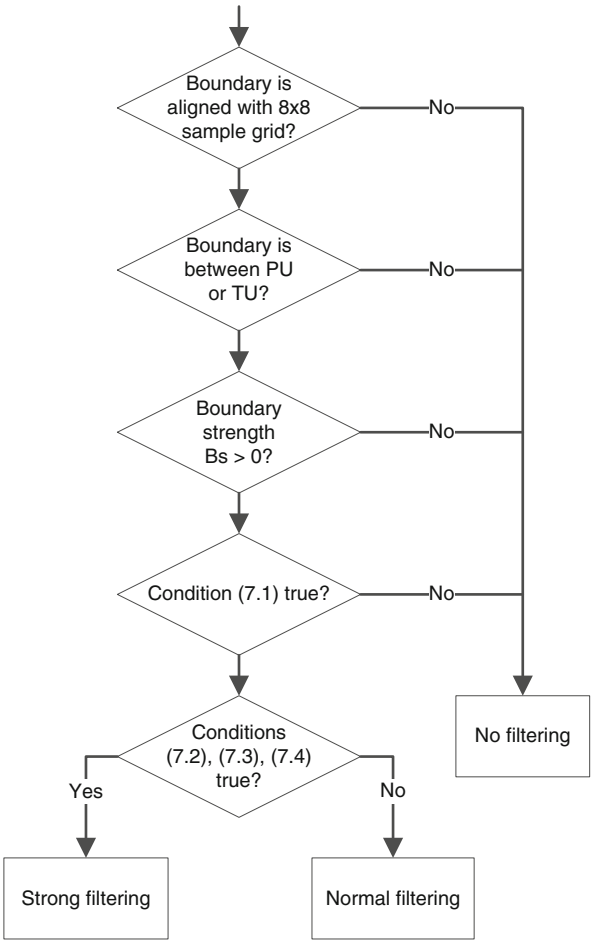


Fig. 7.7 Decisions for each four-sample segment of block boundary. *PU* prediction unit, *TU* transform unit [37]

crossing the block boundary. Since condition in (7.1) is evaluated true for the signal that forms a perfect ramp passing across the block boundary (such as a gradual change in the luma component), the deblocking in the normal filtering mode is designed to not modify the ramp. The filtered sample values p_0' and q_0' (the row index j is omitted for brevity) are determined by adding or subtracting an offset value $\triangle_0$ to each of the sample values:

$$p_0' = p_0 + \Delta_0, \tag{7.7}$$

$$q_0' = q_0 - \Delta_0, \tag{7.8}$$

where the value of $\triangle_0$ is obtained as in

$$\Delta_0 = \text{Clip3}\left(-t_C, t_C, \delta\right), \tag{7.9}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00189

where t_C is a clipping parameter dependent on the QP, and Clip3(a, b, x) function clips the variable x to the range (a, b), i.e.

$$\text{Clip3}\,(a, b, x) = \text{Max}\,(a, \text{Min}\,(b, x))\,, \tag{7.10}$$

and δ is determined as

$$\delta = (9 * (q_0 - p_0) - 3 * (q_1 - p_1) + 8) >> 4. \tag{7.11}$$

Neglecting the clipping operation, the impulse response of the filter is (3, 7, 9, −3)/16. The value of δ is proportional to the deviation of the signal at the sides of the block boundary from a ramp and is equal to zero when the signal across the boundary has the form of a perfect ramp across the block boundary [35].

Deblocking filtering is only applied to a line of samples across the block boundary if the absolute value of δ is below t_C, i.e.

$$|\delta| < 10\, t_C\,. \tag{7.12}$$

Expression (7.12) evaluates whether the discontinuity at the block boundary is likely to be a natural edge or caused by a block artifact.

If two samples are modified in block P, i.e. condition in (7.5) is true, the sample p_1 is modified as

$$p_1{}' = p_1 + \Delta p_1, \tag{7.13}$$

and if condition in (7.6) is true, sample q_1 is modified as

$$q_1{}' = q_1 + \Delta q_1, \tag{7.14}$$

where the $p_1{}'$ and $q_1{}'$ are new values of samples p_1 and q_1 respectively, and the values of Δp_1 and Δq_1 are obtained as follows:

$$\Delta p_1 = \text{Clip3}\,(-t_C/2, t_C/2, ((((p_2 + p_0 + 1) >> 1) - p_1 + \Delta_0) >> 1))\,, \tag{7.15}$$

$$\Delta q_1 = \text{Clip3}\,(-t_C/2, t_C/2, ((((q_2 + q_0 + 1) >> 1) - q_1 - \Delta_0) >> 1))\,. \tag{7.16}$$

The impulse response of the filter is (8, 19, –1, 9, –3)/32 if the clipping operation is neglected. One can see that the value of the offset obtained in (7.9) is used in calculation of Δp_1 and Δq_1. The filtering operations at positions p_0, p_1, q_0, and q_1 do not modify the signal that has a form of a perfect ramp across the block boundary.

The deblocking filter decisions done for each line of a four-sample segment of a block boundary are summarized in a flowchart in Fig. 7.8.

An example of modifications to the block boundary samples in the normal filtering mode is shown in Fig. 7.9.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00190

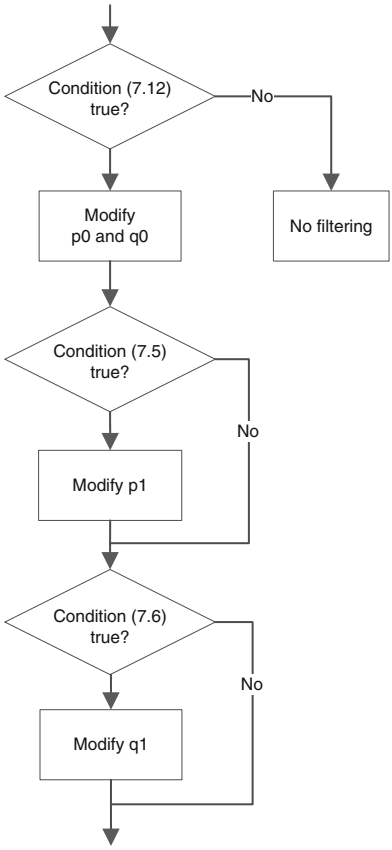


Fig. 7.8 Decisions for normal filter that are applied to each line of four-sample segment of block boundary

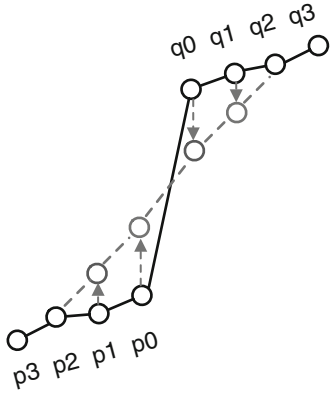


Fig. 7.9 Illustration of normal filtering mode operations: original block boundary (*solid black line*) and modified block boundary (*dashed gray line*)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00191

7.2.2.3 Strong Filtering Mode

The strong deblocking filter in HEVC is applied to smooth flat areas, where block artifacts are more visible. This filtering mode modifies three samples from the block boundary and enables strong low-pass filtering. The HEVC strong filter is similar to the strong filter used by the H.264/AVC video standard [21] except the clipping operation, which is not present in the H.264/AVC strong filter. The reason for the clipping operation in HEVC deblocking filter is that the strong filtering decision is made based on the sample values in only two of the lines in the block, corresponding to $i = 0$ and $i = 3$. The clipping operation limits the amount of filtering in order to make sure that there is no excessive filtering on the lines which were not evaluated in the filtering decisions [19]. The filtering for the samples in block P is performed using the following equations if the clipping operation is neglected:

$$p_0' = (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4) >> 3, \tag{7.17}$$

$$p_1' = (p_2 + p_1 + p_0 + q_0 + 2) >> 2, \tag{7.18}$$

$$p_2' = (2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4) >> 3, \tag{7.19}$$

where p_0', p_1', and p_2' are modified values of samples p_0, p_1, and p_2 respectively. The modified sample values are then clipped to the range $[p_i - 2t_C, p_i + 2t_C]$. The equations for modification of samples q_0, q_1, and q_2 can be obtained by replacing p with q in (7.17)–(7.19).

7.2.2.4 Chroma Deblocking

As mentioned in Sect. 7.2.2.1, the deblocking is only applied to the chroma block boundaries which have boundary strength *Bs* equal to 2, i.e. when one of the adjacent blocks is intra-predicted. The block boundary should also be a CU, TU or a prediction partition boundary and be aligned with the 8×8 chroma sample grid. No further evaluation on the signal is done for chroma block boundaries. Chroma deblocking only modifies one sample from each side of the block boundary. The following expression is used to obtain the modification offset for the chroma block boundary.

$$\Delta_c = \text{Clip3}\left(-t_C, t_C, (((q_0 - p_0) << 2) + p_1 - q_1 + 4) >> 3\right) \tag{7.20}$$

The value of Δ_c is used for modification of the chroma samples p_0 and q_0 similarly to luma samples as in (7.7) and (7.8).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00192

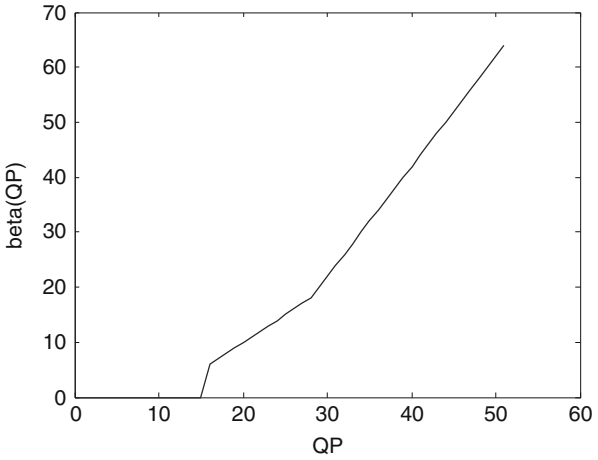


Fig. 7.10 Dependency of β on QP. Reproduced with permission from [37], © 2012 IEEE

7.2.2.5 Deblocking Adaption

Clipping operations are used in deblocking to avoid excessive filtering. The clipping parameter t_C is derived from a look-up table and depends on the average of the quantization parameter QP of the two adjacent blocks [44], which determines how coarse the quantization is. Variable t_C is obtained from a table as t_C (QP) when both adjacent blocks are inter-predicted ($Bs = 1$) and t_C (QP + 2) when at least one of the adjacent blocks is intra-predicted ($Bs = 2$). The deblocking parameter β, which mainly determines which block boundaries are modified by the deblocking filtering, also depends on the quantization parameter QP. One can see the dependency of the parameter β on the QP in Fig. 7.10 and parameter t_C on QP in Fig. 7.11. The higher the QP, the higher are the values of β and t_C and therefore the more often the deblocking filtering is applied, and more samples from the block boundary are modified. In addition, greater modifications to the sample values are allowed for higher QP. For low QP values, when a reconstructed picture has higher quality, the deblocking is essentially turned off by setting β and t_C to zero.

The deblocking strength can be further adjusted on a slice or picture level by sending parameters `tc_offset_div2` and `beta_offset_div2` in the slice header or picture parameter set (PPS) [46]. These parameters specify offsets (divided by two) that are added to a QP value before determining β and t_C from the look-up tables. This gives an encoder a possibility to adapt the deblocking strength depending on the sequence characteristics, the encoding mode, and other factors.

Changing the deblocking parameters on a picture level can be used to improve the subjective quality when using a hierarchical coding structure. An example of a hierarchical-B GOP8 coding structure, similar to the one used in the HM common test conditions random-access mode [6] is shown in Fig. 7.12. In this structure, the encoder can use QP cascading in order to improve the compression

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00193

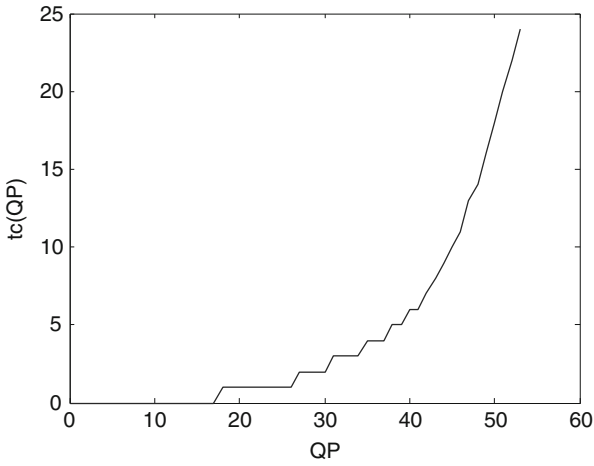


Fig. 7.11 Dependency of t_C on QP. Reproduced with permission from [37], © 2012 IEEE

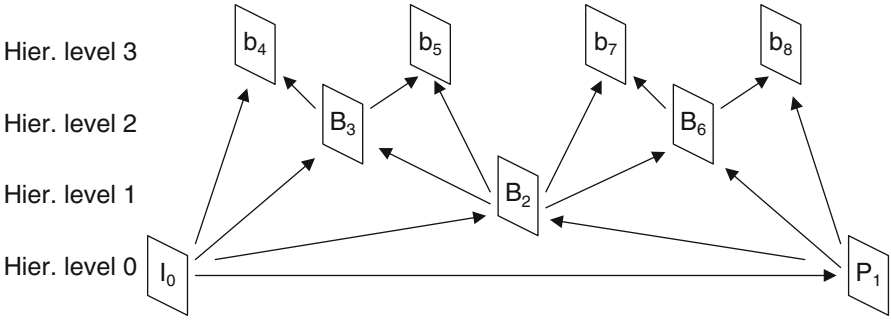


Fig. 7.12 Hierarchical-B coding structure with GOP8 illustrating different depth of the picture in the coding structure [38]

efficiency by applying the base QP to the intra-coded pictures, QP + 1 to the B-pictures at hierarchy level 0 (e.g. picture P_1), QP + 2 to the pictures at hierarchy level 1. Hierarchy level 2 uses QP + 3 and non-reference b-pictures at level 3 use QP + 4. The improvement in compression efficiency is achieved by coding with better quality the pictures at lower hierarchical levels, which are used for prediction of the pictures at higher hierarchy levels. However, in video sequences with chaotic motion, e.g. showing water, smoke, fire, rain, or snow, the QP cascading may cause block artifacts in the pictures at higher hierarchy levels, which is related to using lower bit budget for those pictures because of higher QP values and coarser mode decisions.

When the encoder uses hierarchical coding structure and QP cascading, the deblocking parameters `tc_offset_div2` and `beta_offset_div2` can be used to increase the deblocking strength for the pictures at higher hierarchy levels, which improves the subjective quality on sequences with chaotic motion, while

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00194

preserving the subjective quality on other types of video content [33, 38, 39]. An example of deblocking parameters improving the subjective quality on chaotic content in the hierarchical-B GOP8 coding structure is setting the `tc_offset_div2` to values 0, 3, 3, and 5 for pictures at hierarchy levels 0, 1, 2, and 3 respectively, while the `beta_offset_div2` is set to zero for all levels.

7.3 Sample Adaptive Offset (SAO)

7.3.1 Motivation and Overview of SAO

The key function of the sample adaptive offset is to attenuate ringing artifacts, which are more likely to appear when larger transform sizes are used. The SAO reduces sample distortion by first classifying the samples in the region into multiple categories with as selected classifier and adding a specific offset to each sample depending on its category. The classifier index and the offsets for each region are signaled in the bitstream [7, 8]. The SAO encoder is not standardized. It may minimize the average sample rate-distortion cost, as done in the HM reference software, or may use another criterion to generate SAO parameters. SAO can use different sample offsets in a region depending on the sample classification, and SAO parameters can change from one region of a picture to another. The HEVC uses two SAO types: *edge offset* (EO) and *band offset* (BO) [11]. In EO, the classification of a sample is based on its neighborhood, i.e. on the comparison between the current sample and its neighboring samples. In BO, the classification is based on the sample value.

The best coding efficiency could be achieved by a picture-based region partitioning method [11], which would, however, introduce additional encoding latency. In order to achieve low encoding latency and reduce the buffer requirement, the size of the region can be fixed and set as small as one coding tree unit (CTU). Multiple CTUs can share the same SAO parameters by region merging [14] to reduce side information. In HEVC, a CTU consists of three coding tree blocks (CTBs) of color components, and each color component can have its own SAO offsets and share the same EO/BO type for chroma components [10].

7.3.2 Edge Offset

Figure 7.13 shows the Gibbs phenomenon, which can be used to explain the appearance of ringing artifacts in image and video coding. The horizontal axis shows the sample position along a 1-D line and the vertical axis denotes the sample value. The dotted curve represents the original samples while the solid curve represents the reconstructed samples when the highest frequencies in the signal are discarded due to quantization of transform coefficients. Local peaks, convex edges/corners,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00195

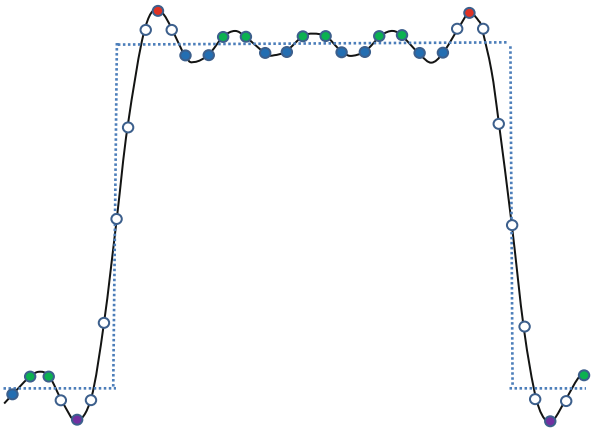


Fig. 7.13 Gibbs phenomenon, where the *dotted curve* is the original samples and the *solid curve* is the reconstructed samples. Local peaks, convex corners, concave corners, and local valleys are marked with *solid circles* and none-of-the-above samples with *empty circles*. Reproduced with permission from [13], © 2012 IEEE

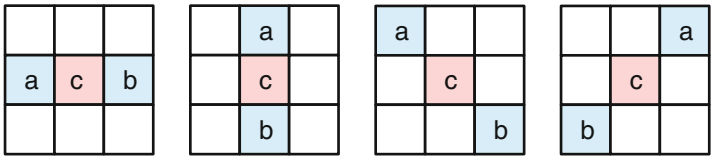


Fig. 7.14 Four 1-D directional patterns for EO sample classification: horizontal (EO class = 0), vertical (EO class = 1), 135° diagonal (EO class = 2), and 45° diagonal (EO class = 3). Reproduced with permission from [13], © 2012 IEEE

concave edges/corners, and local valleys are marked with solid circles and none-of-the-above samples with empty circles. One can observe from the figure that the distortion can be reduced by applying negative offsets to local peaks and convex corners and positive offsets to concave corners and valleys. The operation of the edge offset is based on this observation. EO uses four one-directional patterns for sample classification: horizontal, vertical, 135° diagonal, and 45° diagonal, as shown in Fig. 7.14, where label "c" represents the current sample and labels "a" and "b" represent the two neighboring samples. These four sample patterns form four EO classes. Only one EO class can be selected for each CTB that enables EO. Based on the rate-distortion optimization, one EO class is chosen and an index indicating which EO class is selected is signaled in the bitstream.

For a given EO class with the specific direction, each sample inside the CTB is classified into one of five categories. The current sample value, labeled as "c", is compared with its two neighbors along the selected 1-D pattern. The category classification rules for each sample are summarized in Table 7.2. Categories 1 and 4 are associated with a local valley and a local peak, respectively, along the selected 1-D pattern. Categories 2 and 3 are associated with concave and convex corners, respectively. If the current sample does not belong to any of EO categories 1–4, it is assigned to category 0 and SAO is not applied. Note that the categories are mutually exclusive and a sample can belong only to one category.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00196

Table 7.2 Sample category classification rules for edge offset

Category	Condition
1	c < a && c < b
2	(c < a && c==b) || (c == a && c < b)
3	(c > a && c==b) || (c == a && c > b)
4	c > a && c > b
0	None of the above

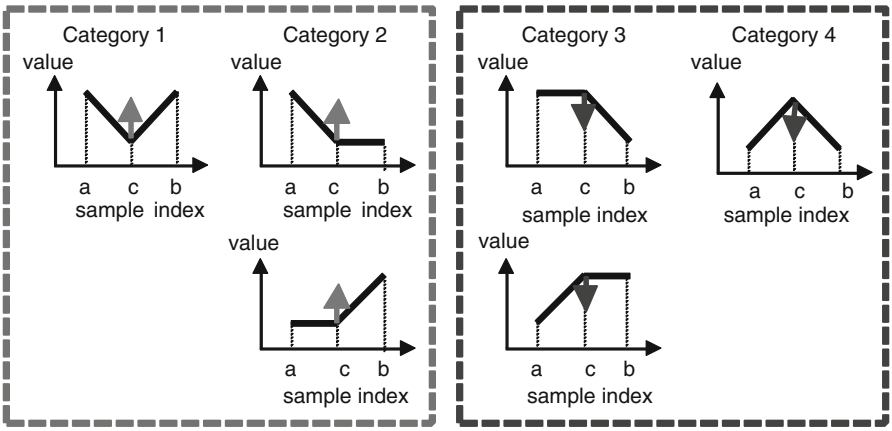


Fig. 7.15 Positive offsets for EO categories 1 and 2 and negative offsets for EO categories 3 and 4 results in smoothing, where the x-axis is sample index and the y-axis is sample value. Reproduced with permission from [13], © 2012 IEEE

The effect of the positive and negative edge offsets is illustrated in Fig. 7.15 and explained as follows. Positive offsets for categories 1 and 2 result in smoothing since local valleys and concave corners become smoother, while negative offsets for these categories result in sharpening. On the contrary, for categories 3 and 4, the negative offsets result in smoothing and positive offsets result in sharpening. In HEVC, sharpening in EO is not allowed. Therefore, the absolute values of four specific offsets are signaled by the encoder—one for each EO category, and the signs of the signaled offsets are implicitly derived from the corresponding EO categories [12, 23, 24]. Both EO and BO use four offsets, which limits the number of offsets to reduce the requirements for a line buffer (the line buffer is explained further in Sect. 7.4.3).

7.3.3 Band Offset

Another offset used by the HEVC SAO tool is band offset (BO). In band offset, one offset is added to all samples whose values belong to the same band (range of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00197

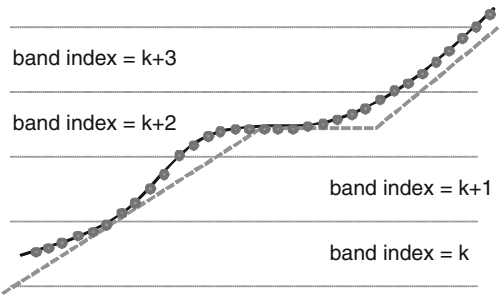


Fig. 7.16 Example of BO, where the *dotted curve* represents original samples and the *solid curve* denotes reconstructed samples. Reproduced with permission from [13], © 2012 IEEE

values). The sample values range is divided into 32 equal bands. For 8-bit samples in the range from 0 to 255, the width of a band is 8. Thus, sample values from $8k$ to $8k + 7$ belong to band k, where k ranges from 0 to 31. The difference between the original samples and reconstructed samples in a band (i.e. the offset of a band) can be signaled to the decoder. There is no constraint on the offset sign for the BO.

Figure 7.16 demonstrates how the BO compensates sample intensity offset of a region. The horizontal axis denotes the sample position and the vertical axis denotes the sample value. The dotted curve represents the original samples, while the solid curve denotes the reconstructed samples, affected by quantization errors of prediction residues and phase shifts because of the coded motion vectors that deviate from the true motion. As shown in Fig. 7.16, if there is a phase shift (difference) between the reconstructed motion vector and the "true" motion vector, a smooth region with a gradient may be offset with a certain value compared to the original signal. In this example, the reconstructed samples are shifted to the left compared to the original samples, which results in a systematic negative error that can be corrected by BO for bands k, $k + 1$, $k + 2$, and $k + 3$, where the samples ranging from $k * 8$ to $((k + 1) * 8) - 1$ are classified as belonging to band k, and can be modified by using the corresponding offset value.

In HEVC, only offsets of four consecutive bands and the starting (or minimum) band position of the current region are signaled to the decoder [25, 29]. Four offsets are signaled in the BO, which is equal to the number of signaled offsets in EO (the number of offsets is limited to reduce the line buffer requirement). The reason for signaling only four bands is that the range of sample values in a region formed by CTBs can be quite limited. Therefore, by signaling the starting band position of current region, BO can identify the minimum sample value to be compensated in the current region so that the decoder can recover it, as shown in the example in Fig. 7.17. This is especially true for chroma CTBs. In natural images, chroma components are often represented by a narrow-band signal, which means that by several band offsets, the encoder can recover most samples in the region.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00198

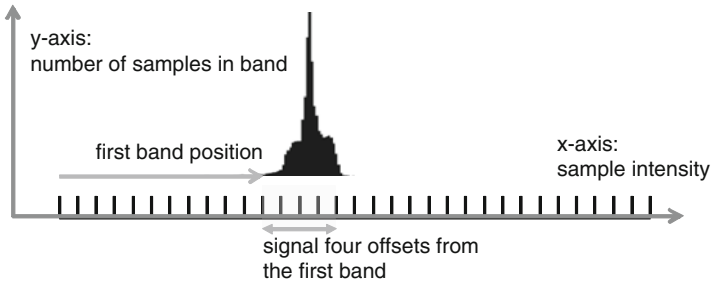


Fig. 7.17 Example of sample distribution in a CTB, where BO sends the offsets of four consecutive bands. Reproduced with permission from [13], © 2012 IEEE

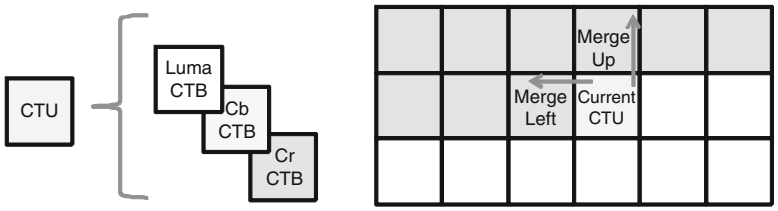


Fig. 7.18 CTU consists of three CTBs of color components; the current CTU can reuse SAO parameters of the left or above CTU. Reproduced with permission from [13], © 2012 IEEE

7.3.4 SAO Parameters Signaling

A syntax element `sample_adaptive_offset_enabled_flag` signaled in the Sequence Parameter Set (SPS) indicates whether SAO is enabled in the current video sequence. In the slice header, two syntax elements, `slice_sao_luma_flag` and `slice_sao_chroma_flag`, indicate if SAO is enabled for luma and chroma, respectively, in the current slice.

Low-delay applications can use the Coding Tree Unit (CTU) based SAO encoding algorithm. As shown in Fig. 7.18, a CTU comprises its corresponding luma CTB, Cb CTB, and Cr CTB. Syntax-wise, the basic unit for SAO parameters adaptation is always one CTU. If SAO is enabled in the current slice, the SAO parameters of each CTU are interleaved into the slice data. The SAO data in the bitstream are signaled in the beginning of each CTU. The CTU-level SAO parameters consist of SAO merging information, type information, and offset information.

7.3.4.1 SAO Parameters Merging

A CTU can use three options for signaling SAO parameters: reusing SAO parameters of the left CTU (by setting a syntax element `sao_merge_left_flag`

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00199

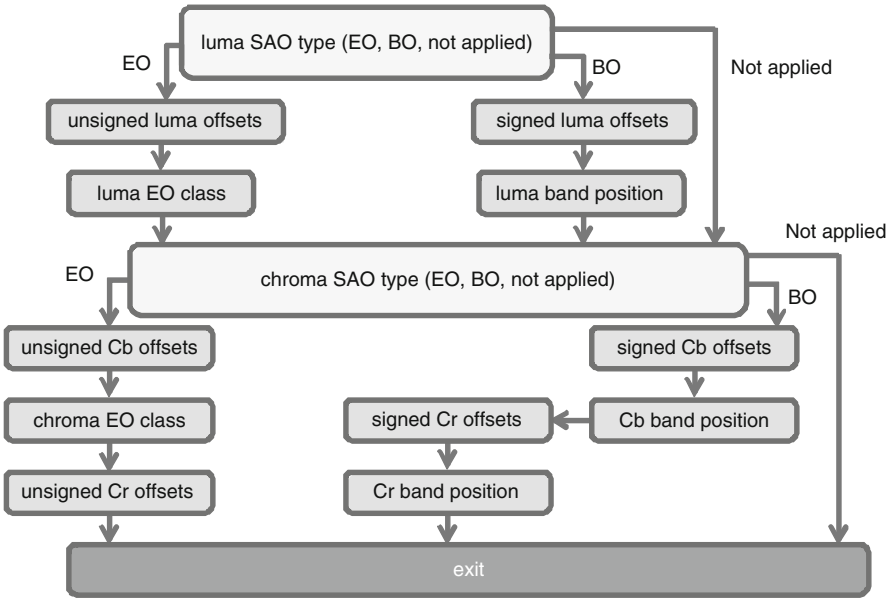


Fig. 7.19 Illustration of CTU-level SAO information coding when the current CTU is not merged with the CTU on the left or above. Reproduced with permission from [13], © 2012 IEEE

to 1), reusing SAO parameters of the above CTU (by setting a syntax element `sao_merge_up_flag` to 1), or by transmitting new SAO parameters. The SAO merging information is shared by all three color components. When SAO merge-left or SAO merge-up mode is indicated, all SAO parameters from the left or above CTU are copied and no more information is signaled for the current CTU. This CTU-based SAO information merging effectively reduces the SAO information that needs to be signaled [31].

7.3.4.2 SAO Type and Offsets Signaling

If merging of SAO information is not used, the information for the current CTU is signaled as shown in Fig. 7.19. Syntax elements for the luma component are first sent, followed by the Cb syntax elements and then the Cr syntax elements. For each color component, the SAO type is transmitted (`sao_type_idx_luma` or `sao_type_idx_chroma`), which indicates EO, BO, or not applied (SAO turned off). If BO or EO is selected, four offsets are transmitted. If BO is selected, the starting band position (`sao_band_position`) is signaled. Otherwise, if EO is selected, the EO class (`sao_eo_class_luma` or `sao_eo_class_chroma`) is signaled. The Cb and Cr components share the SAO type (`sao_type_idx_chroma`) and EO class (`sao_eo_class_chroma`)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00200

syntax elements to reduce the side information and speed-up SAO processing by achieving more efficient memory access on certain platforms [4]. These syntax elements are therefore only coded for the Cb component. The design of the codewords (including "off", "EO class selection index", and "BO band position index") is based on the probability distribution to reduce side information.

7.3.4.3 CABAC Contexts and Bypass Coding

All CTU-level, SAO syntax elements including SAO merging information, SAO type information, and offset information are coded with context-based adaptive binary arithmetic coding (CABAC). Only the first bin of the SAO type, which specifies whether SAO is turned on or off in the current CTU, and the SAO merge-left and merge-up flags use CABAC contexts. All other bins are coded in the bypass mode, which significantly increases the SAO parsing throughput in CABAC without much coding efficiency loss [1, 3, 15, 30, 45].

7.4 Implementation and Parallelization Aspects

7.4.1 Deblocking Filter Complexity and Parallelism

When designing the HEVC deblocking filter, a lot of attention was paid to complexity and parallelization aspects. In H.264/AVC video decoders, deblocking takes a significant part of the computational complexity [27, 28]. Moreover, in H.264/AVC, the deblocking operations at one block boundary may affect the samples used in deblocking of the next block boundary, which complicates parallel processing.

7.4.1.1 HEVC Deblocking Filter Complexity

The complexity of the HEVC deblocking filter has been significantly decreased compared to the H.264/AVC deblocking. First of all, the deblocking is only applied to the block boundaries on the 8×8 luma sample grid. This already decreases the worst-case complexity of the deblocking compared to H.264/AVC, where the deblocking is applied on the 4×4 sample grid. The average complexity of the deblocking operation is also decreased compared to H.264/AVC since the prediction and transform blocks in HEVC are on average larger than those in H.264/AVC, where the maximum size of prediction blocks is 16×16 luma samples and the size of the transform blocks is 8×8 luma samples (if the maximum transform size in HEVC is not restricted to the same limits resulting in the worst-case scenario).

The filtering decisions constitute a significant part of the deblocking filter complexity. In order to reduce the complexity of deblocking decisions, the HEVC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00201

deblocking uses decisions for a four-sample segment of the block boundary based on two lines crossing the block boundary. In contrast, the decisions in H.264/AVC deblocking are done for every line. The complexity of chroma deblocking filtering in HEVC has also been reduced compared to H.264/AVC since only chroma block boundaries with *Bs* equal to 2 are filtered. Therefore, only block boundaries adjacent to the intra-predicted blocks are filtered in the chroma components in contrast to H.264/AVC, where the chroma deblocking is also applied to the block boundaries between the inter-predicted blocks.

7.4.1.2 Deblocking Filter Parallelization Aspects

HEVC deblocking filter allows easy parallelization on several levels. First, parallelization is possible on the color component level. In HEVC, filtering decisions for chroma components are only based on the block boundary strength. Therefore, the only data to be shared between the luma and the chroma deblocking is the *Bs*, which depends on the prediction type [43]. This makes it possible to process chroma components independently of the luma component unlike in H.264/AVC, where chroma deblocking uses the decisions made for luma deblocking.

The vertical and horizontal block boundaries in HEVC are processed in a different order than in H.264/AVC. In HEVC, all the vertical block boundaries in the picture are filtered first, and then all the horizontal block boundaries are filtered [20, 37]. Since the minimum distance between two parallel block boundaries in HEVC is eight samples, and HEVC deblocking modifies at most three samples from the block boundary and uses four samples from the block boundary for deblocking decisions, filtering of one vertical boundary does not affect filtering of any other vertical boundary. This means there are no deblocking dependencies across the block boundaries. In principle, any vertical block boundary can be processed in parallel to any other vertical boundary. The same holds for the horizontal boundaries, although the modified samples from filtering the vertical boundaries are used as the input to filtering the horizontal boundaries.

The deblocking in HEVC can also be performed on the 8×8 block basis [32, 37, 47]. Figure 7.20 illustrates how the deblocking (both for vertical and horizontal boundaries) can be performed independently for each 8×8 block of samples. The deblocking is performed on the 8×8 luma sample grid and decisions are done separately for each four-sample segment of the block boundary, which means that two parts of the eight-sample block boundary are deblocked independently of each other [34]. Therefore, the deblocking of the 8×8 sample square with the crossing of vertical and horizontal lines on the 8×8 sample filtering grid in the middle of the block is not dependent on the deblocking in the other parts of the picture. Basically, the whole picture can be split into such 8×8 sample blocks (4×8 and 8×4 blocks at the picture boundaries), which can all be processed independently of other blocks. Since all vertical block boundaries in HEVC are processed before the horizontal

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00202

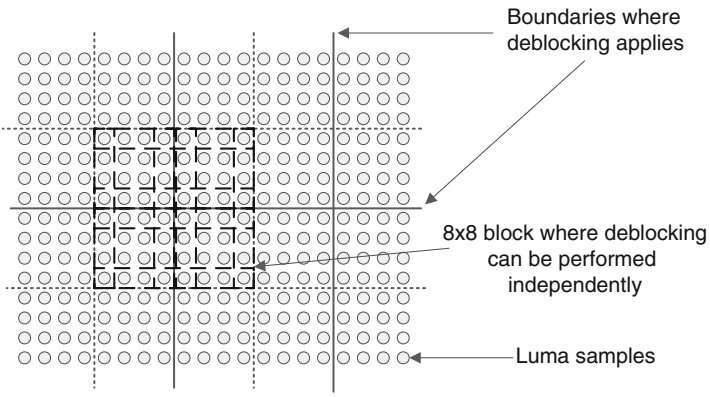


Fig. 7.20 Illustration of picture samples, horizontal and vertical block boundaries on the 8×8 grid, and those non-overlapping blocks of 8×8 samples (marked with *dotted lines*), which can be deblocked in parallel. The *dashed lines* mark samples used in deblocking decisions (vertical and horizontal)

block boundaries, the order of deblocking in each of these 8×8 deblocking units is the same: the vertical block boundary is filtered first, which is followed by the horizontal block boundary.

Since the HEVC deblocking can be easily parallelized, it can be done on a slice or tile [16] basis. In this case, an encoder or decoder can choose the option to first apply deblocking to the inner areas of a tile or slice, while leaving the deblocking on the tile or slice boundaries. When the decoding and deblocking of all tiles or slices is finished, the tile or slice boundaries can be processed as the last step.

Since the deblocking in HEVC is less computationally expensive and more parallelizable than the H.264/AVC deblocking, it can be said that the deblocking in HEVC has a better trade-off between the computational complexity, throughput, subjective and objective quality improvements than the H.264/AVC deblocking and is less of a bottleneck when implementing a decoder.

7.4.2 SAO Implementation Aspects and Parameters Estimation

Since SAO requires sample level operations to classify each sample into bands or categories in both encoder and decoder, the number of operations for each sample needs be reduced as much as possible to reduce the overall computational complexity. At encoder-side, there are many SAO types to be tested to achieve a better rate-distortion performance at reasonable computational complexity. Therefore, some efficient encoder algorithms are discussed in the following sections.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00203

7.4.2.1 Fast Edge Offset Sample Classification

Although the sample classification rules of EO in Table 7.2 seem non-trivial, the EO sample classification can be implemented in a more efficient way by using the following function and equations:

$$sign3(x) = (x > 0)\ ?\ (+1) : (x == 0)\ ?\ (0) : -1, \tag{7.21}$$

$$edgeIdx = 2 + sign3\,(c - a) + sign3\,(c - b)\,, \tag{7.22}$$

$$edgeIdx2category\,[\,] = \{1, 2, 0, 3, 4\}\,, \tag{7.23}$$

$$category = edgeIdx2category\,[edgeIdx] \tag{7.24}$$

where, "c" is the current sample, and "a" and "b" are the two neighboring samples, as shown in Fig. 7.14 and Table 7.2. As a further speed-up, the data obtained in a previous step can be reused in the classification of the next sample. For example, assume that the EO class is 0 (i.e., a 1-D horizontal pattern) and the samples in the CTB are processed in the raster scan order. The "$sign3(c-a)$" of the current sample is equal to "$-sign3(c-b)$" of the neighboring sample to the left. Likewise, the "$sign3(c-b)$" of the current sample can be reused by the neighboring sample to the right. In software implementations, the $sign3(x)$ function can be implemented by using a bitwise operation or a look-up table to avoid using if-else operation, which can be time-consuming on certain platforms.

7.4.2.2 Fast Band Offset Sample Classification

The sample range is equally divided into 32 bands in BO. Since 32 is equal to two to the power of five, the BO sample classification can be implemented as using the five most significant bits of each sample as the classification result. In this way, the complexity of BO decreases, especially in hardware that only needs wire connections without logic gates to obtain the classification result from the sample value. To reduce the software decoding run time, the BO classification can be implemented by using bitwise operation or a look-up table to avoid using if-else operations.

7.4.2.3 Distortion Estimation for Encoder

The rate-distortion optimization process [41] requires multiple calculation of the distortion between the original and reconstructed sample values. A straightforward SAO implementation would add offsets to the samples modified by deblocking and then calculate the distortion between the resulting and the original samples. To reduce the memory access and the number of operations, a fast distortion estimation method [9] can be implemented as follows. Let k, $s(k)$, and $x(k)$ be

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00204

sample positions, original samples, and the reconstructed samples, respectively, where k belongs to C, the set of samples inside the CTB that belong to a specific SAO type (i.e., BO or EO), a starting band position or EO class, and a specific band or category. The distortion between original samples and reconstructed samples can be described by the following equation:

$$D_{pre} = \sum_{k \in C} (s(k) - x(k))^2 \tag{7.25}$$

The distortion between the original samples and samples modified by SAO can be described by the following equation

$$D_{post} = \sum_{k \in C} (s(k) - (x(k) + h))^2 \tag{7.26}$$

where h is the offset for the sample set. The distortion change is defined by the following equation:

$$\Delta D = D_{post} - D_{pre} = \sum_{k \in C} \left(h^2 - 2h\,(s(k) - x(k))\right) = Nh^2 - 2hE \tag{7.27}$$

where N is the number of samples in the set, and E is the sum of differences between the original samples and the reconstructed samples (before SAO) as defined by the following equation:

$$E = \sum_{k \in C} (s(k) - x(k)) \tag{7.28}$$

Please note that the sample classification and (7.28) can be calculated right after the input samples become available after the deblocking filtering. Thus, N and E can be calculated only once and stored. Then, the delta rate-distortion cost is defined as follows:

$$\Delta J = \Delta D + \lambda R \tag{7.29}$$

where λ is the Lagrange multiplier, and R represents the estimated bits of side information.

For a given CTB with a specific SAO type (i.e., BO or EO), starting band position or EO class, and a specific band or category, several h values (offsets) close to E/N are tested, and the offset that minimizes ΔJ is chosen. After offsets for all bands or categories have been chosen, the ΔJ for each of the 32 bands of BO or each of the five categories of EO are added to obtain the delta (change) of the rate-distortion cost of the entire CTB. The distortion of the BO bands using zero offsets and the EO category 0 can be pre-calculated by (7.25) and stored for subsequent re-use. When SAO decreases the cost for the entire CTB (i.e. the delta cost is negative), SAO is enabled for this CTB. Similarly, the best SAO type and the best starting position or EO class can be found by the fast distortion estimation.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00205

7.4.2.4 Slice-Level On/Off Control

The HM reference software common test conditions [6] use hierarchical quantization parameter (QP) settings. As an example, in the random access condition, the GOP size is eight. Picture can belong to different hierarchy levels depending on their positions in the GOP (see Fig. 7.12). Normally, the picture is only predicted from the pictures with a smaller or the same hierarchy level. A picture with at higher hierarchy level will likely be given a higher QP.

A slice-level on/off decision algorithm [2, 26] is provided as follows. For hierarchy level 0 pictures, SAO is always enabled in the slice header. Given a current picture with a nonzero hierarchy level N, the previous picture is defined as the last picture with hierarchy level $(N-1)$ in the decoding order. If SAO is disabled in more than 75 % of the CTBs in the previous picture, the HM reference encoder will disable SAO in all slice headers of the current picture and skip the SAO encoding process. This encoder technique not only can decrease the number of syntax to be parsed, but also provides 0.5 % BD-rate improvement [2, 26]. Please note that luma and chroma SAO can be enabled or disabled independently in the slice header.

7.4.2.5 SAO Parameters Estimation and Interaction with Deblocking

In the HM reference encoder, SAO parameters are estimated for each CTU. Since SAO is applied to the output of the deblocking filter, the SAO parameters cannot be precisely determined until the deblocked samples are available. However, the deblocked samples of the right columns and the bottom rows in the current coded-tree block (CTB) may be unavailable because the CTU to the right and the CTU below the current CTU may not have been reconstructed yet (in a one-pass encoder). This constraint can be overcome with one of the two options. The first option [18] estimates the SAO parameters on the available CTB samples, i.e. on the CTB samples except the three bottom rows of luma samples, one bottom row of Cb and Cr samples, the rightmost four columns of luma samples, the rightmost two columns of Cb and Cr samples. The proposed approach does not incur a noticeable coding efficiency loss when the 64×64 CTU size is used. However, for smaller CTU sizes, the percentage of samples not used in the SAO parameter estimation is higher, which may cause significant coding efficiency loss. In this case, the second option [22] uses samples before the deblocking instead of unavailable deblocked samples during SAO parameter estimation, which can reduce the loss in coding efficiency for smaller CTU sizes.

7.4.3 CTU-Based Processing and Line Buffer

CTU-based processing is commonly adopted in practical implementations. CTUs are encoded or decoded one by one in a raster scan order and the in-loop filtering is applied right after the encoding/decoding of a CTU. The deblocking filtering of the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00206

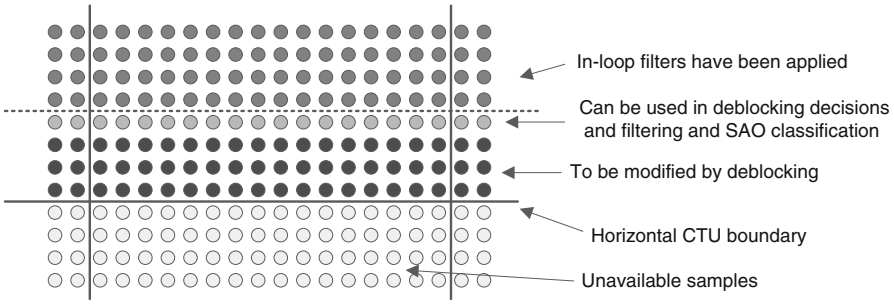


Fig. 7.21 Example of luma samples line buffer. Samples below the *dashed line* and above the horizontal CTU boundary should be kept in the line buffer until the unavailable samples have been reconstructed

bottom horizontal CTU boundary needs samples from the CTU below. Hence, after the current CTU has been processed, the deblocking cannot be applied yet to the bottom rows of samples since the reconstructed samples of the CTB to the bottom of the current CTB are not yet available (see Fig. 7.21). Likewise, since SAO is applied after the deblocking filter, SAO cannot be applied to the bottom sample rows before the deblocking is done. In order to decrease memory bandwidth requirements, the information needed for in-loop filtering over the lower CTU boundary is kept in the fast on-chip memory until the CTU below has been reconstructed and the in-loop filters applied. This on-chip memory is usually called a "line buffer" since the information for horizontal lines of samples typically needs to be kept.

In the deblocking filter, vertical filtering across a horizontal CTU boundary needs four rows of luma samples, two rows of Cb samples, and two rows of Cr samples from the upper CTU to be kept in the line buffer for the filtering decisions and operations. Deblocking can modify up to three rows of luma samples, one row of Cb samples, and one row of Cr samples from the block boundary.

Let us assume that the deblocking filter needs to keep N rows of the above CTBs, where N is equal to four for luma and two for Cb and Cr. Since the N-th row above the horizontal CTB boundary will not to be modified by the vertical deblocking filtering, the SAO can be applied to the $(N+1)$-th row above the horizontal CTB boundary. However, the bottom N rows of the current CTB should be kept in the line buffer before the CTB below is reconstructed and deblocking and SAO are applied (see Fig. 7.21 for luma samples example).

When the CTB below is reconstructed and SAO in the CTB above uses EO with the class greater than zero, applying SAO for the N-th row above the boundary needs the $(N+1)$-th row above the boundary to be available. A straightforward solution is to store reconstructed and deblocked samples of the $(N+1)$-th row in the line buffer. However, using the fast EO sample classification, the "*sign3*" results [the sign of the difference or 0 value between the N-th row and the $(N+1)$-th row] can be stored instead, which reduces the memory requirements to two bits per sample.

In addition to the described four lines on luma samples, two lines of Cb, two lines of Cr samples, three lines of two-bit "*sign3*" values, and some CTU- and PU-level information need to be stored. The deblocking needs information about

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00207

Table 7.3 Size of the elements of a 16 × 16 CTU needed to be kept in a line buffer

Values in line buffer	Required bits
Four motion vectors*	128 bits
4 reference picture indices*	16 bits
8 × 8 / 4 × 8 partitions flags*	2 bits
B-/P-prediction flags in 8 × 8 partitions*	2 bits
sign3	64 bits
Luma SAO type	2 bits
Chroma SAO type	2 bits
Starting band positions or EO classes	15 bits
Luma and chroma offsets	48 bits
Total elements	**279 bits**
64 luma samples	512 bits
32 chroma samples	256 bits
Total samples	**768 bits**
Total bits for CTU	**1047 bits**

*This information is also used in motion vector derivation

the motion vectors adjacent to the lower CTU boundary, and SAO needs SAO type and SAO offsets of the upper CTB row. For an 8-bit 4:2:0 video and the smallest CTU size (16 × 16 luma and 8 × 8 chroma samples), the information stored per CTU is 279 bits (this number may depend on the implementation). Among these 279 bits, 128 bits are for four motion vectors (4 × 8 and 8 × 4 partitions can only use one motion vector), 16 bits are for four indices of reference pictures in the decoded picture buffer, two bits are for signaling whether 4 × 8 or 8 × 8 partitions are used, and two bits are for signaling if one or two MVs are used in 8 × 8 partitions. Please note that this information is also needed for other modules, such as motion vector derivation. Alternatively, 8 bits with four Bs values may be stored if the information about motion vectors in the next CTU row is available. Then, 64 bits are for "*sign3*" values of 16 luma and 16 chroma samples, two bits are for luma SAO type, two bits are for chroma SAO type, 15 bits are from starting band positions or EO classes, and 48 bits are from luma and chroma offsets. The sample line buffers for the 16 × 16 CTU keep 64 luma samples and 32 chroma samples, which requires 768 bits (see Table 7.3 for details). Therefore, the total line buffer size is about 15K bytes for full HD (i.e. 1920 × 1080) videos.

The line buffer size would be somewhat smaller when larger CTU sizes are used since the same parameters apply to a larger number of samples. The size of the line buffer can be further reduced by using vertical tiles, hence splitting the horizontal "span" of CTUs to be processed before the bottom CTU is encoded/decoded.

7.4.4 Error Resilience

In order to provide additional error resilience, HEVC allows an encoder to disable in-loop filters over tile and slice boundaries. A flag `slice_loop_`

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00208

`filter_across_slices_enabled_flag` equal to 1 indicates that the in-loop filters are applied across the left and upper boundaries of the current slice. When the flag is equal to 0, in-loop filtering operations are not applied across the left and upper boundaries of the current slice. When the flag is not present, it is inferred to be equal to `pps_loop_filter_across_slices_enabled_flag`.

A picture parameters set (PPS) flag `loop_filter_across_tiles_enabled_flag` equal to 1 specifies that in-loop filters are applied across tile boundaries. When the flag is equal to 0, it specifies that in-loop filtering operations are not performed across tile boundaries. When the flag is not present, the value of `loop_filter_across_tiles_enabled_flag` is inferred to be equal to 1.

These flags provide additional error resilience since an error created because of a slice or tile loss will not propagate into neighboring slices or tiles of the same picture (however, it may propagate to other slices or tiles of subsequent pictures because of inter-picture prediction).

7.5 Coding Efficiency and Subjective Quality Improvements

The HEVC in-loop filters improve both the objective and subjective quality. The objective quality improvement is achieved due to increasing the quality of the reconstructed pictures. There is also an additional effect due to better quality of reference pictures, which improves motion prediction and therefore the coding efficiency.

In this section, the compression efficiency improvements have been evaluated on the JCT-VC video sequences test set. The results are provided for several classes of sequences and under different coding conditions defined in the HEVC common test conditions document [6]. These configurations are: All Intra where only intra-prediction is used; Random Access which uses intra pictures over certain time intervals and hierarchical-B coding structure; and two low-delay configurations, which have only one intra-picture, and where motion-compensated prediction uses only temporally preceding pictures. The Low Delay P (LP) configuration does not use bi-directional motion-compensated prediction. The BD-rate is used in the HEVC standardization as a measure for the average bit rate reduction at the same mean squared error [5]. The HEVC reference software HM11.0 was used in all experiments. The reported decoding time has been evaluated by decoding the bitstreams on a Windows 7 (64bit) PC with i7-920 CPU and 8GB of RAM without writing the reconstructed pictures to the disk.

7.5.1 Deblocking Coding Efficiency and Subjective Quality Improvements

The results for objective performance of the deblocking filter are provided in Table 7.4. The results show that applying HEVC deblocking leads to the 1.3–3.2 %

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00209

Table 7.4 Luma BD-rates evaluating objective effects of using deblocking filtering under various coding conditions

Anchor: disable deblocking Test: enable deblocking		Y BD-rate			
		All Intra (AI)	Random Access (RA)	Low Delay B (LB)	Low Delay P (LP)
Class A	Traffic	−2.3 %	−2.7 %	n/a	n/a
Cropped 4K × 2K	PeopleOnStreet	−2.0 %	−4.4 %	n/a	n/a
	Nebuta	−1.1 %	−1.1 %	n/a	n/a
	SteamLocomotive	−2.3 %	−5.4 %	n/a	n/a
Class B	Kimono	−4.1 %	−5.7 %	−5.9 %	−8.0 %
1080p	ParkScene	−1.4 %	−1.9 %	−2.0 %	−2.8 %
	Cactus	−1.3 %	−3.4 %	−3.7 %	−4.5 %
	BasketballDrive	−1.8 %	−3.9 %	−3.7 %	−4.9 %
	BQTerrace	−0.3 %	−1.1 %	−0.6 %	−2.3 %
Class C	BasketballDrill	−0.7 %	−2.1 %	−2.1 %	−2.8 %
WVGA	BQMall	−1.5 %	−2.5 %	−2.6 %	−3.1 %
	PartyScene	−0.4 %	−0.8 %	−0.8 %	−0.9 %
	RaceHorses	−1.2 %	−2.6 %	−2.9 %	−3.2 %
Class D	BasketballPass	−1.4 %	−2.5 %	−2.5 %	−2.9 %
WQVGA	BQSquare	−0.1 %	0.0 %	0.6 %	0.3 %
	BlowingBubbles	−0.4 %	−0.9 %	−0.8 %	−1.1 %
	RaceHorses	−0.9 %	−2.3 %	−2.4 %	−2.7 %
Class E	FourPeople	−2.3 %	n/a	−4.9 %	−6.6 %
720p	Johnny	−2.2 %	n/a	−2.2 %	−5.8 %
	KristenAndSara	−2.1 %	n/a	−4.0 %	−6.0 %
Class F	BasketballDrillText	−0.6 %	−2.0 %	−1.9 %	−2.5 %
	ChinaSpeed	−0.7 %	−1.8 %	−1.9 %	−2.3 %
	SlideEditing	−0.3 %	−0.3 %	−0.5 %	−0.7 %
	SlideShow	−0.8 %	−0.9 %	−0.8 %	−1.1 %
Class summary	Class A	−1.9 %	−3.4 %	n/a	n/a
	Class B	−1.8 %	−3.2 %	−3.2 %	−4.5 %
	Class C	−0.9 %	−2.0 %	−2.1 %	−2.5 %
	Class D	−0.7 %	−1.4 %	−1.3 %	−1.6 %
	Class E	−2.2 %	n/a	−3.7 %	−6.1 %
	Class F	−0.6 %	−1.2 %	−1.3 %	−1.6 %
Overall summary	All	−1.3 %	−2.3 %	−2.3 %	−3.2 %
	Decoding time (%)	107 %	104 %	104 %	105 %

bit rate decrease at the same quality depending on the configuration. For certain classes of sequences, 6 % decrease of bit rate is achieved.

Figure 7.22 compares a part of a decoded picture from the *BasketballDrive* sequence (1080p, 50 fps) in Random Access configuration at QP = 32 where the deblocking was applied with the configuration where the deblocking was turned off. Figure 7.23 shows same comparison for the sequence *KristenAndSara*

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00210



Fig. 7.22 Sequence *BasketballDrive*, Random Access, QP32: (**a**) deblocking turned off, (**b**) deblocking turned on



Fig. 7.23 Sequence *KristenAndSara*, Low Delay, QP37: (**a**) deblocking turned off, (**b**) deblocking turned on

(720p@60 fps) coded in Low Delay B configuration at QP = 37. It can be seen that the deblocking filter effectively attenuates block artifacts.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00211

7.5.2 SAO Coding Efficiency and Subjective Quality Improvement

This section illustrates the subjective and objective performance of the SAO tool. Table 7.5 reports the sequence-wise luma BD-rates and the average luma BD-rates and run-times for different encoding structures and CTU sizes equal to 64×64 in luma using the skipping boundary samples algorithm as described in Sect. 7.4.2.5. For *BQTerrace* in the LP condition, the SAO coding gain reaches 18.9 %. It is noted that SAO is particularly effective for Class F sequences, which mostly contain computer graphics and screen content rather than natural video. One could also notice that SAO shows higher coding gains in the LP configuration without bi-directional prediction. Regarding the computational complexity, SAO increases the average decoding time by less than 2–3 %.

The subjective quality improvements due to reduction of ringing artifacts are shown in Figs. 7.24 and 7.25. Figure 7.24 shows an example of the coded computer-generated sequence SlideEditing. SAO significantly improves visual quality by suppressing ringing *artifacts* near objects edges. Figure 7.25 shows examples of natural video sequences *RaceHorses* and *BasketballPass* where the edges of objects are much cleaner when SAO is enabled. According to viewing tests, SAO improves subjective quality [42].

7.5.3 Combined Effect of In-Loop Filters on Coding Efficiency

Table 7.6 demonstrates objective compression efficiency improvements due to both in-loop filters compared to the configuration where both the deblocking filter and SAO are turned off. One can see that the compression efficiency improvements are 2.6–15 % depending on coding configurations. The decoding time increase is about 10 % and depends on the coding conditions. The encoding complexity mostly depends on a particular encoder implementation and is not significant in the HM11.0 encoder operating in common test conditions (on the order of 1 % encoding time increase [13]). These numbers indicate that the in-loop filters are an efficient tool in improving the HEVC compression efficiency.

7.6 Main Differences between HEVC and H.264/AVC In-Loop Filters

This section summarizes key differences between the HEVC and H.264/AVC in-loop filters. There is only a deblocking in-loop filter in H.264/AVC, while the HEVC standard defines two in-loop filters: the deblocking filter and the sample adaptive offset, SAO.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00212

Table 7.5 Luma BD-rates evaluating objective effects of using SAO under various coding conditions

Anchor: disabling SAO Test: enabling SAO CTU Size in luma: 64 × 64 CTU boundary: option 1		Y BD-rate			
		All Intra (AI)	Random Access (RA)	Low Delay B (LB)	Low Delay P (LP)
Class A	Traffic	−0.9 %	−1.2 %	n/a	n/a
Cropped 4K × 2K	PeopleOnStreet	−1.3 %	−2.1 %	n/a	n/a
	Nebuta	−0.1 %	−2.3 %	n/a	n/a
	SteamLocomotive	−0.2 %	−2.6 %	n/a	n/a
Class B	Kimono	−0.5 %	−0.6 %	−0.8 %	−7.8 %
1080p	ParkScene	−0.7 %	−0.8 %	−1.3 %	−9.1 %
	Cactus	−0.4 %	−2.4 %	−2.9 %	−10.4 %
	BasketballDrive	−0.2 %	−1.5 %	−1.5 %	−9.0 %
	BQTerrace	−0.5 %	−4.8 %	−3.8 %	−18.9 %
Class C	BasketballDrill	−1.0 %	−1.7 %	−2.7 %	−6.4 %
WVGA	BQMall	−0.3 %	−0.9 %	−1.7 %	−8.2 %
	PartyScene	−0.1 %	0.2 %	−0.7 %	−4.0 %
	RaceHorses	−0.5 %	−1.9 %	−1.8 %	−9.7 %
Class D	BasketballPass	−0.2 %	−0.6 %	−1.3 %	−4.7 %
WQVGA	BQSquare	−0.5 %	−0.0 %	−0.7 %	−3.9 %
	BlowingBubbles	−0.2 %	0.4 %	0.0 %	−2.9 %
	RaceHorses	−0.5 %	−1.2 %	−1.3 %	−6.4 %
Class E	FourPeople	−0.7 %	n/a	−2.9 %	−9.1 %
720p	Johnny	−0.4 %	n/a	−1.9 %	−13.3 %
	KristenAndSara	−0.6 %	n/a	−2.3 %	−11.4 %
Class F	BasketballDrillText	−1.1 %	−2.0 %	−3.9 %	−6.7 %
	ChinaSpeed	−1.3 %	−3.6 %	−6.9 %	−9.3 %
	SlideEditing	−1.5 %	−3.0 %	−4.5 %	−5.0 %
	SlideShow	−1.9 %	−2.2 %	−5.4 %	−6.4 %
Class summary	Class A	−0.6 %	−2.0 %	n/a	n/a
	Class B	−0.4 %	−2.0 %	−2.0 %	−11.1 %
	Class C	−0.5 %	−1.1 %	−1.7 %	−7.1 %
	Class D	−0.4 %	−0.3 %	−0.8 %	−4.5 %
	Class E	−0.6 %	n/a	−2.6 %	−11.3 %
	Class F	−1.5 %	−2.7 %	−5.2 %	−6.8 %
Overall summary	All	−0.7 %	−1.7 %	−2.4 %	−9.2 %
	Decoding time (%)	103 %	102 %	102 %	103 %

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00213

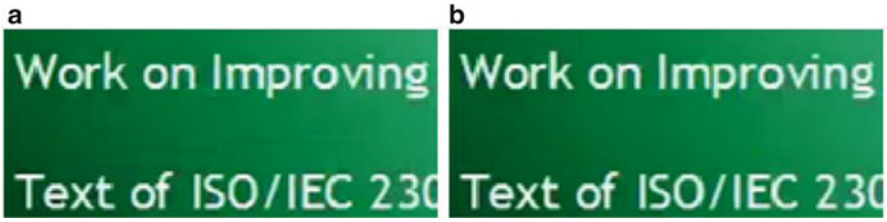


Fig. 7.24 Example of test sequence *SliceEditing* in LP condition, POC = 100, QP = 32: (**a**) SAO is disabled, (**b**) SAO is enabled

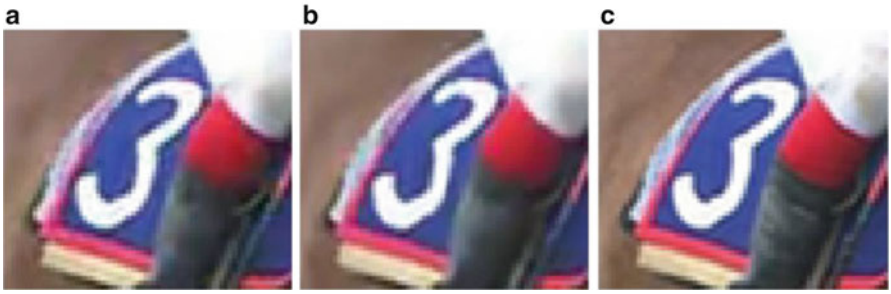


Fig. 7.25 Subjective quality comparison of *RaceHorses* test sequence, POC = 20, QP = 32, LP condition: (**a**) SAO is disabled, (**b**) SAO is enabled, (**c**) original (uncoded) sequence

The computational complexity of the HEVC deblocking is lower than that of the H.264/AVC. Reduction of the HEVC deblocking complexity is achieved by restricting the filtering to the 8×8 sample grid in contrast to the 4×4 sample grid in the H.264/AVC deblocking. Additional complexity reduction in HEVC is achieved by making the sample-based filtering decisions based on a subset of lines across the block boundary in contrast to the line-based decisions in H.264/AVC deblocking. Moreover, the HEVC deblocking of chroma components is only applied to the intra-predicted block boundaries. The HEVC deblocking is also more suitable for parallel implementation than the H.264/AVC deblocking since each 8×8 sample block in HEVC can be deblocked independently of other 8×8 blocks and the order of the vertical and horizontal filtering operations in HEVC deblocking is always the same. The processing order for the horizontal and vertical block boundaries is therefore different in HEVC and H.264/AVC. When applying the HEVC deblocking on the CU basis, right after the CU reconstruction, filtering of four right-most samples of horizontal block boundaries in a CU should be delayed until the next CU to the right is reconstructed and the vertical boundary between the CUs is filtered.

HEVC and H.264/AVC deblocking filters are also different in terms of criteria that evaluate the signal (reconstructed sample values) at the sides of a block boundary to decide whether the deblocking is applied to this block boundary. In H.264/AVC, the deblocking is typically applied to the block boundary when the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00214

Table 7.6 Luma BD-rates evaluating the objective effects of using deblocking filter and SAO under various coding conditions

Anchor: disable deblocking and SAO Test: enable deblocking and SAO		Y BD-rate			
		All Intra (AI)	Random Access (RA)	Low Delay B (LB)	Low Delay P (LP)
Class A	Traffic	−4.4 %	−5.0 %	n/a	n/a
Cropped 4K × 2K	PeopleOnStreet	−4.4 %	−8.1 %	n/a	n/a
	Nebuta	−1.5 %	−4.9 %	n/a	n/a
	SteamLocomotive	−3.3 %	−9.5 %	n/a	n/a
Class B	Kimono	−6.1 %	−7.7 %	−8.1 %	−22.0 %
1080p	ParkScene	−2.9 %	−3.5 %	−4.2 %	−18.1 %
	Cactus	−2.3 %	−6.7 %	−7.2 %	−18.9 %
	BasketballDrive	−2.7 %	−6.5 %	−6.1 %	−18.2 %
	BQTerrace	−1.0 %	−6.7 %	−5.3 %	−25.7 %
Class C	BasketballDrill	−2.4 %	−4.5 %	−5.4 %	−11.9 %
WVGA	BQMall	−2.3 %	−4.0 %	−5.2 %	−15.2 %
	PartyScene	−0.7 %	−1.0 %	−2.0 %	−7.7 %
	RaceHorses	−2.4 %	−5.8 %	−5.9 %	−16.0 %
Class D	BasketballPass	−2.3 %	−3.7 %	−4.8 %	−10.4 %
WQVGA	BQSquare	−0.7 %	−0.1 %	−0.3 %	−5.2 %
	BlowingBubbles	−0.9 %	−0.7 %	−1.2 %	−6.3 %
	RaceHorses	−2.1 %	−4.4 %	−4.9 %	−12.0 %
Class E	FourPeople	−3.8 %	n/a	−8.4 %	−20.6 %
720p	Johnny	−3.4 %	n/a	−5.3 %	−28.2 %
	KristenAndSara	−3.4 %	n/a	−6.9 %	−23.9 %
Class F	BasketballDrillText	−2.3 %	−4.6 %	−6.2 %	−11.6 %
	ChinaSpeed	−2.3 %	−5.9 %	−9.6 %	−13.2 %
	SlideEditing	−1.8 %	−3.4 %	−5.5 %	−5.6 %
	SlideShow	−2.8 %	−3.3 %	−6.9 %	−8.4 %
Class summary	Class A	−3.4 %	−6.9 %	n/a	n/a
	Class B	−3.0 %	−6.2 %	−6.2 %	−20.6 %
	Class C	−2.0 %	−3.8 %	−4.6 %	−12.7 %
	Class D	−1.5 %	−2.2 %	−2.8 %	−8.5 %
	Class E	−3.5 %	n/a	−6.9 %	−24.2 %
	Class F	−2.3 %	−4.3 %	−7.1 %	−9.7 %
Overall summary	All	−2.6 %	−4.8 %	−5.5 %	−15.0 %
	Decoding time (%)	113 %	107 %	107 %	109 %

signal on both sides of the boundary is flat. Therefore in HEVC, the deblocking is also applied when the signal on each side of the block boundary approximates a ramp or a slope, which can happen in smooth areas with changing luma intensity.

The SAO in-loop filter attenuates ringing artifacts, which can be more pronounced in HEVC when larger transform sizes are used by the encoder. Moreover, SAO can also be applied to the inside samples of the large blocks, which cannot be corrected by the deblocking filter. This is especially important for HEVC because of the large transform sizes allowed in the standard.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00215

In case of a CTU-based processing, four lines of samples for the luma component and two lines for the chroma components should be kept in a line buffer for in a line buffer both the HEVC and H.264/AVC deblocking. The fourth line of samples would also be modified by SAO in HEVC and is therefore delayed to be written to the memory compared to H.264/AVC. Some additional memory is required for keeping the SAO parameters (see Table 7.3).

7.7 Conclusions

HEVC defines two in-loop filters, deblocking and sample adaptive offset (SAO), which significantly improve the subjective quality of decoded video sequences as well as compression efficiency by increasing the quality of the reconstructed/reference pictures. The deblocking filter attenuates discontinuities on the block boundaries, while SAO mainly corrects ringing artifacts caused by large transforms and quantization and sample value offsets in certain regions of a picture caused by coding of motion vectors. The complexity of the HEVC deblocking has been significantly reduced compared to the H.264/AVC. Moreover, the HEVC deblocking filter is highly parallelizable with parallelization down to 8×8 sample blocks. Having lower computational complexity and being highly parallelizable, the HEVC deblocking is less of a bottleneck in the decoder implementation than the H.264/AVC deblocking and provides better trade-off between the computational complexity and coding efficiency (subjective and objective quality). SAO is a new in-loop filter, not present in H.264/AVC, which provides significant reduction of ringing artifacts at relatively low decoding complexity. The deblocking and SAO can also be implemented in the same processing unit, which simplifies CTU-based encoding and decoding and reduces cost in hardware implementations.

References

1. Alshina E, Alshin A, Park JH (2012) AHG5: on bypass coding for SAO syntax elements, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0043, Stockholm, July 2012
2. Alshina E, Alshin A, Park JH (2012) Encoder modification for SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0044, Stockholm, July 2012
3. Alshina E, Alshin A, Park JH, Fu C-M, Huang Y-W, Lei S (2012) AHG5/AHG6: on reducing context models for SAO merge syntax, Joint Collaborative Team on Video Coding Coding (JCT-VC), Document JCTVC-J0041, Stockholm, July 2012
4. Alshina E, Alshin A, Park JH, Laroche G, Gisquet C, Onno P (2012) AHG6: on SAO type sharing between U and V components, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0045, Stockholm, July 2012
5. Bjøntegaard G (2001) Calculation of average PSNR differences between RD-curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00216

6. Bossen F (2013) Common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L1100, Geneva, Jan. 2013
7. Fu C-M, Chen C-Y, Huang Y-W, Lei S (2010) TE10 Subtest 3: Quadtree-based adaptive offset, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C147, Guangzhou, Oct. 2010
8. Fu C-M, C-Y Chen, Huang Y-W, Lei S (2011) CE8 Subset 3: picture quadtree adaptive offset, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D122, Daegu, Jan. 2011
9. Fu C-M, Chen C-Y, Huang Y-W, Lei S (2011) Sample adaptive offset for HEVC. In: IEEE 13th international workshop on multimedia signal processing (MMSP) 2011
10. Fu C-M, Chen C-Y, Huang Y-W, Lei S, Park S, Jeon B, Alshin A, Alshina E (2011) Sample adaptive offset for chroma, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F057, Torino, July 2011
11. Fu C-M, Chen C-Y, Tsai C-Y, Huang Y-W, Lei S (2011) CE13: sample adaptive offset with LCU-independent decoding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E049, Geneva, Mar. 2011
12. Fu C-M, Huang Y-W, Lei S, Chong IS, Karczewicz M (2011) Non-CE8: offset coding in SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G222, Geneva, Nov. 2011
13. Fu C-M, Alshina E, Alshin A, Huang Y-W, Chen C-Y, Tsai C-Y, Hsu C-W, Lei S, Park JH, Han W-J (2012) Sample adaptive offset in the HEVC standard. IEEE Trans Circuits Syst Video Technol 22(12):1755–1764
14. Fu C-M, Chen C-Y, Tsai C-Y, Huang Y-W, Lei S, Chong IS, Karczewicz M, Alshina E, Alshin A (2012) E8.a.3: SAO with LCU-based syntax, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0273, San Jose, Feb. 2012
15. Fu C-M, Huang Y-W, Lei S (2012) Non-CE1: bug-fix of offset coding in SAO interleaving mode, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0168, Geneva, Apr. 2012
16. Fuldseth A, Horowitz M, Xu S, Segall A, Zhou M (2011) Tiles, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F335, Torino, July 2011
17. Han W-J, Min J, Kim IK, Alshina E, Alshin A, Lee T, Chen J, Seregin V, Lee S, Hong YM, Cheon MS, Sklyakhov N, McCann K, Davies T, Park JH (2010) Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools. IEEE Trans Circuits Syst Video Technol 20(12):1709–1720
18. Huang Y-W, Alshina E, Chong IS, Wan W, Zhou M (2012) Description of core experiment 1 (CE1): sample adaptive offset filtering, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H1101, San Jose, Feb. 2012
19. Ikeda M, Suzuki T (2012) Non-CE10: introduction of strong filter, Joint Collaborative Team on Video Coding, Document JCTVC-H0275, San Jose, Feb. 2012
20. Ikeda M, Tanaka J, Suzuki T (2011) CE12 Subset2: parallel deblocking filter, Joint Collaborative Team on video coding (JCT-VC), Document JCTVC-E181, Geneva, Mar. 2011
21. ITU-T Rec. H.264 and ISO/IEC 14496-10 (2003) Advanced Video Coding
22. Kim W-S (2012) AHG6: SAO parameter estimation using non-deblocked pixels, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0139, Stockholm, July 2012
23. Kim W-S, Kwon D-K (2012) Non-CE8: method of visual coding artifact removal for SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G680, Geneva, Nov. 2012
24. Kim W-S, Kwon D-K (2012) CE8 Subset c: necessity of sign bits for SAO offsets, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0434, San Jose, Feb. 2012
25. Laroche G, Poirier T, Onno P (2011) On additional SAO band offset classifications, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G246, Geneva, Nov. 2011
26. Laroche G, Poirier T, Onno P (2012) Non-CE1: encoder modification for SAO interleaving mode, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0184, Geneva, Apr. 2012

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00217

27. List P, Josh A, Lainema J, Bjøntegaard G, Karczewicz M (2003) Adaptive loop filter. IEEE Trans Circuits Syst Video Technol 13:614 -619
28. Lou J, Jagmohan A, He D, Lu L, Sun M-T (2009) H.264 deblocking speedup. IEEE Trans Circuits Syst Video Technol 19(8):1178–1182
29. Maani E, Nakagami O (2012) Flexible band offset mode in SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0406, San Jose, Feb. 2012
30. Minezawa A, Sugimoto K, Sekiguchi S (2012) Non-CE1: improved edge offset coding for SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0066, Geneva, Apr. 2012
31. Minoo K, Baylon D (2012) AHG6: coding of SAO merge left and merge up flags, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0355, Stockholm, July 2012
32. Narroschke M, Esenlik S, Wedi T (2011) CE12 Subtest 1: results for modified decisions for deblocking, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G590, Geneva, Nov. 2011
33. Norkin A (2012) Non-CE1: non-normative improvement to deblocking filtering, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-K0289, Shanghai, Oct. 2012
34. Norkin A (2012) CE10.3: deblocking filter simplifications: Bs computation and strong filtering decision, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0473, San Jose, Feb. 2012
35. Norkin A, Andersson K, Sjöberg R, Huang Q, An J, Guo X, Lei S (2011) CE12: Ericsson's and MediaTek's deblocking filter, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F118, Torino, July 2011
36. Norkin A, Andersson K, Fuldseth A, Bjøntegaard G (2012) HEVC deblocking filtering and decisions. In: *Proc. SPIE.* 8499, Applications of Digital Image Processing XXXV, no. 849912, Oct. 2012
37. Norkin A, Bjøntegaard G, Fuldseth A, Narroschke M, Ikeda M, Andersson K, Zhou M, Van der Auwera G (2012) HEVC deblocking filter. IEEE Trans Circuits Syst Video Technol 22(11):1746–1754
38. Norkin A, Andersson K, Kulyk V (2013) Two HEVC encoder methods for block artifact reduction. In: Proceedings of the IEEE international conference on visual communications and image processing (VCIP) 2013, Kuching, Sarawak, 17–20 Nov. 2013
39. Norkin A, Andersson K, Sjöberg R (2013) AHG6: on deblocking filter and parameters signaling, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L0232, Geneva, Jan. 2013
40. ITU-T Rec. H.265 and ISO/IEC 23008-2 (2013) High efficiency video coding
41. Sullivan GJ, Wiegand T (1998) Rate-distortion optimization for video compression. IEEE Signal Processing Magazine, pp 74–90
42. Tan TK, Fujibayashi A, Suzuki Y, Takiue J (2012) AHG8: objective and subjective evaluation of HM5.0, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0116, San Jose, Feb. 2012
43. Ugur K, Andersson KR, Fuldseth A (2010) Video coding technology proposal by Tandberg, Nokia, and Ericsson, Joint Collaborative Team on Video Coding, Document JCTVC-A119, Dresden, Apr. 2010
44. Van der Auwera G, Wang X, Karczewicz M, Narroschke M, Kotra A, Wedi T (2011) Support of varying QP in deblocking, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1031, Geneva, Nov. 2011
45. Xu J, Tabatabai A (2012) AHG6: on SAO signaling, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0268, Stockholm, July 2012
46. Yamakage T, Asaka S, Chujoh T, Karczewicz M, Chong IS (2011) CE12: deblocking filter parameter adjustment in slice level, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G174, Geneva, Nov. 2011
47. Zhou M, Sezer O, Sze V (2011) CE12 subset 2: test results and architectural study on deblocking filter without parallel on/off filter decision, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G088, Geneva, Nov. 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00218

Chapter 8
Entropy Coding in HEVC

Vivienne Sze and Detlev Marpe

Abstract Context-Based Adaptive Binary Arithmetic Coding (CABAC) is a method of entropy coding first introduced in H.264/AVC and now used in the latest High Efficiency Video Coding (HEVC) standard. While it provides high coding efficiency, the data dependencies in H.264/AVC CABAC make it challenging to parallelize and thus limit its throughput. Accordingly, during the standardization of entropy coding for HEVC, both aspects of coding efficiency and throughput were considered. This chapter describes the functionality and design methodology behind CABAC entropy coding in HEVC.

8.1 Introduction

Context-Based Adaptive Binary Arithmetic Coding (CABAC) [46] is a form of entropy coding used in H.264/AVC [63] and also in HEVC [64]. Entropy coding is a lossless compression scheme that uses the statistical properties to compress data such that the number of bits used to represent the data is logarithmically proportional to the probability of the data. For instance, when compressing a string of characters, frequently used characters are each represented by a few bits, while infrequently used characters are each represented by many bits. From Shannon's information theory [72], when the compressed data is represented in bits {0,1}, the optimal average code length for a character with probability p is $-\log_2 p$.

V. Sze (✉)
Massachusetts Institute of Technology (MIT), Cambridge, MA, USA
e-mail: sze@mit.edu

D. Marpe
Fraunhofer Institute for Telecommunications Heinrich Hertz Institute (HHI), Berlin, Germany
e-mail: Detlev.Marpe@hhi.fraunhofer.de

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_8
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00219

Entropy coding is performed at the last stage of video encoding (and first stage of video decoding), after the video signal has been reduced to a series of syntax elements. Syntax elements describe how the video signal can be reconstructed at the decoder. This includes the method of prediction (e.g., spatial or temporal prediction) along with its associated prediction parameters as well as the prediction error signal, also referred to as the residual signal. Note that in HEVC only the syntax elements belonging to the slice segment data are CABAC encoded. All other high level syntax elements are coded either with zero-order Exponential (Exp)-Golomb codes or fixed-pattern bit strings. Table 8.1 shows the syntax elements that are encoded with CABAC in HEVC and H.264/AVC. For HEVC, these syntax elements describe properties of the coding tree unit (CTU), prediction unit (PU), and transform unit (TU), while for H.264/AVC, the equivalent syntax elements have been grouped together along the same categories in Table 8.1. For a CTU, the related syntax elements describe the block partitioning of the CTU into coding units (CU), whether the CU is intra-picture (i.e., spatially) predicted or inter-picture (i.e., temporally) predicted, the quantization parameters of the CU, and the type (edge or band) and offsets for sample adaptive offset (SAO) in-loop filtering performed on the CTU. For a PU, the syntax elements describe the intra prediction mode or the motion data. For a TU, the syntax elements describe the residual signal in terms of frequency position, sign and magnitude of the quantized transform coefficients.

This chapter describes how CABAC entropy coding has evolved from H.264/AVC to HEVC. While high coding efficiency is important for reducing the transmission and storage cost of video, processing speed and area cost also need to be considered in the development of HEVC in order to handle the demand for higher resolutions and frame rates in future video coding systems. Accordingly, both coding efficiency and throughput improvement tools are discussed. Section 8.2 provides an overview of CABAC entropy coding. Section 8.3 explains the design considerations and techniques used to address both coding efficiency and throughput requirements. Sections 8.4–8.7 describe how these techniques were applied to coding tree unit coding, prediction unit coding, transform unit coding and context initialization, respectively. Section 8.8 compares the coding efficiency, throughput and memory requirements of HEVC and H.264/AVC for both common conditions and worst case conditions.

8.2 CABAC Overview

The CABAC algorithm was originally developed within the joint H.264/AVC standardization process of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). In a first preliminary version, the new entropy-coding method of CABAC was introduced as a standard contribution [44] to the ITU-T VCEG meeting in January 2001. CABAC was adopted as one of two alternative methods of entropy coding within the H.264/AVC standard. The other method specified in H.264/AVC was a low-complexity entropy coding technique

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00220

Table 8.1 CABAC coded syntax elements in HEVC and H.264/AVC

		HEVC	H.264/AVC
Coding Tree Unit (CTU) and Coding Unit (CU) **[Sect. 4]**	Coding Block and Prediction Block Structure, and Quantization Parameters **[Sect. 4.1–4.4]**	split_cu_flag, pred_mode_flag, part_mode, pcm_flag, cu_transquant_bypass_flag, cu_skip_flag, cu_qp_delta_abs, cu_qp_delta_sign_flag, end_of_slice_segment_flag, end_of_sub_stream_one_bit	mb_type, sub_mb_type, mb_skip_flag, mb_qp_delta, end_of_slice_flag, mb_field_decoding_flag
	Sample Adaptive Offset (SAO) Parameters **[Sect. 4.5]**	sao_merge_left_flag, sao_merge_up_flag, sao_type_idx_luma, sao_type_idx_chroma, sao_offset_abs, sao_offset_sign, sao_band_position, sao_eo_class_luma, sao_eo_class_chroma	n/a
Prediction Unit (PU) **[Sect. 5]**	Intra Prediction Mode Data **[Sect. 5.2]**	prev_intra_luma_pred_flag, mpm_idx, rem_intra_luma_pred_mode, intra_chroma_pred_mode	prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag, rem_intra4x4_pred_mode, rem_intra8x8_pred_mode, intra_chroma_pred_mode
	Motion Data **[Sect. 5.1]**	merge_flag, merge_idx, inter_pred_idc, ref_idx_l0, ref_idx_l1, abs_mvd_greater0_flag, abs_mvd_greater1_flag, abs_mvd_minus2, mvd_sign_flag, mvp_l0_flag, mvp_l1_flag	ref_idx_l0, ref_idx_l1, mvd_l0, mvd_l1
Transform Unit (TU) **[Sect. 6]**	Residual Data **[Sect. 6.1, 6.2, 6.3, 6.4, 6.5]**	rqt_root_cbf, split_transform_flag, cbf_luma, cbf_cb, cbf_cr, transform_skip_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix, coded_sub_block_flag, sig_coeff_flag, coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag, coeff_abs_level_remaining, coeff_sign_flag	coded_block_flag, coded_block_pattern, transform_size_8x8_flag, significant_coeff_flag, last_significant_coeff_flag, coeff_abs_level_minus1, coeff_sign_flag

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00221

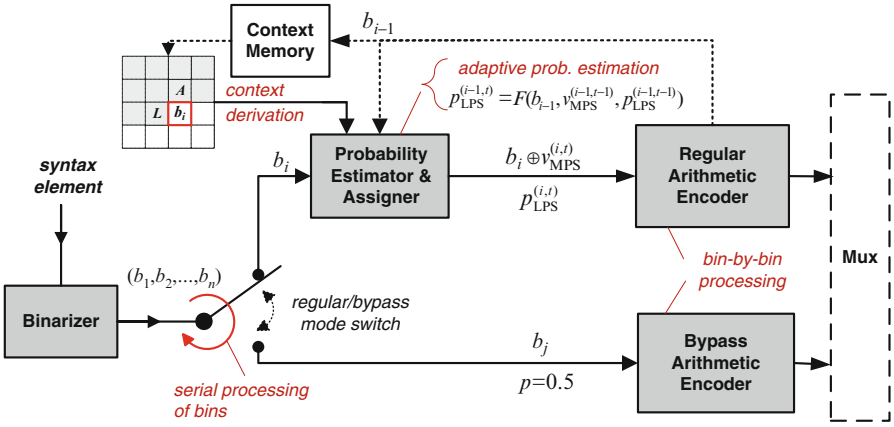


Fig. 8.1 CABAC block diagram (from the encoder perspective): Binarization, context modeling (including probability estimation and assignment), and binary arithmetic coding. In *red*: Potential throughput bottlenecks, as further discussed from the decoder perspective in Sect. 8.3.2

based on the usage of context-adaptively switched sets of variable-length codes, so-called Context-Adaptive Variable-Length Coding (CAVLC). Compared to CABAC, CAVLC offers reduced implementation cost at the price of lower compression efficiency. Typically, the bit-rate overhead for CAVLC relative to CABAC is in the range of 10–16 % for standard definition (SD) interlaced material, encoded at Main Profile, and 15–22 % for high definition (HD) 1080p material, encoded at High Profile, both measured at the same objective video quality and for the case that all other used coding tools within the corresponding H.264/AVC Profile remain the same [46, 48].

CABAC became also part of the first HEVC test model HM1.0 [53] together with the so-called low-complexity entropy coding (LCEC) as a follow-up of CAVLC. Later, during the HEVC standardization process, it turned out that to improve the compression efficiency of LCEC, the complexity of LCEC had to be increased to a point where LCEC was not significantly lower complexity than CABAC. Thus, CABAC in its improved form, both with respect to throughput speed and compression efficiency, became the single entropy coding method of the HEVC standard.

The basic design of CABAC involves the key elements of binarization, context modeling, and binary arithmetic coding. These elements are illustrated as the main algorithmic building blocks of the CABAC encoding block diagram in Fig. 8.1. Binarization maps the syntax elements to binary symbols (bins). Context modeling estimates the probability of each non-bypassed (i.e., regular coded) bin based on some specific context. Finally, binary arithmetic coding compresses the bins to bits according to the estimated probability.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00222

Table 8.2 Examples of different binarizations

		Truncated Unary (TrU)	Truncated Rice (TRk)	Exp-Golomb (EGk)	Fixed-Length (FL)
N	Unary (U)	cMax=7	$k = 1$; cMax=7	$k = 0$	cMax=7
0	0	0	00	1	000
1	10	10	01	010	001
2	110	110	100	011	010
3	1110	1110	101	00100	011
4	11110	11110	1100	00101	100
5	111110	111110	1101	00110	101
6	1111110	1111110	1110	00111	110
7	11111110	1111111	1111	0001000	111

8.2.1 Binarization

The coding strategy of CABAC is based on the finding that a very efficient coding of non-binary syntax element values in a hybrid block-based video coder, like components of motion vector differences or transform coefficient level values, can be achieved by employing a binarization scheme as a kind of preprocessing unit for the subsequent stages of context modeling and arithmetic coding. In general, a binarization scheme defines a unique mapping of syntax element values to sequences of binary symbols, so-called bins, which can also be interpreted in terms of a binary code tree. The design of binarization schemes in CABAC both for H.264/AVC and HEVC is based on a few elementary prototypes whose structure enables fast implementations and which are representatives of some suitable model-probability distributions.

Several different binarization processes are used in HEVC including k-th order truncated Rice (TRk), k-th order Exp-Golomb (EGk), and fixed-length (FL) binarization. Parts of these forms of binarization, including the truncated unary (TrU) scheme as the zero-order TRk binarization, were also used in H.264/AVC. These various methods of binarization can be explained in terms of how they would signal an unsigned value N. Examples are also provided in Table 8.2.

- Unary coding involves signaling a bin string of length $N + 1$, where the first N bins are 1 and the last bin is 0. The decoder searches for a 0 to determine when the syntax element is complete. For the TrU scheme, truncation is invoked for the largest possible value cMax[1] of the syntax element being decoded.
- k-th order truncated Rice is a parameterized Rice code that is composed of a prefix and a suffix. The prefix is a truncated unary string of value $N >> k$, where the largest possible value is cMax. The suffix is a fixed length binary representation of the least significant bins of N; k indicates the number of least

[1] cMax is defined by the standard for each relevant type of syntax element.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00223

significant bins. Note that for k = 0, the truncated Rice is equal to the truncated unary binarization.
- k-th order Exp-Golomb code is proved to be a robust, near-optimal prefix-free code for geometrically distributed sources with unknown or varying distribution parameter. Each codeword consists of a unary prefix of length $l_N + 1$ and a suffix of length $l_N + k$, where $l_N = \lfloor \log_2((N >> k) + 1) \rfloor$ [46].
- Fixed-length code uses a fixed-length bin string with length $\lceil \log_2(\text{cMax} + 1) \rceil$ and with most significant bins signaled before least significant bins.

The binarization process is selected based on the type of syntax element. In some cases, binarization also depends on the value of a previously processed syntax element (e.g., binarization of `coeff_abs_level_remaining` depends on the previously decoded coefficient levels) or slice parameters that indicate if certain modes are enabled (e.g., binarization of partition mode, so-called `part_mode`, depends on whether asymmetric motion partition is enabled). The majority of the syntax elements use the binarization processes as listed above, or some combination of them (e.g., `cu_qp_delta_abs` uses TrU (prefix) + EG0 (suffix) [98]). However, certain syntax elements (e.g., `part_mode` and `intra_chroma_pred_mode`) use custom binarization processes.

During the HEVC standardization process, special attention has been put on the development of an adequately designed binarization scheme for absolute values of transform coefficient levels. In order to guarantee a sufficiently high throughput, the goal here was the maximization of bypass-coded bins under the constraint of not sacrificing coding efficiency too much. This was accomplished by making the binarization scheme adaptive based on previously coded transform coefficient levels. More details on that are given in Sect. 8.6.5.

8.2.2 Context Modeling, Probability Estimation and Assignment

By decomposing each non-binary syntax element value into a sequence of bins, further processing of each bin value in CABAC depends on the associated coding-mode decision, which can be either chosen as the regular or the bypass mode (as described in Sect. 8.2.3). The latter is chosen for bins, which are assumed to be uniformly distributed and for which, consequently, the whole regular binary arithmetic encoding (and decoding) process is simply bypassed. In the regular coding mode, each bin value is encoded by using the regular binary arithmetic coding engine, where the associated probability model is either determined by a fixed choice, based on the type of syntax element and the bin position or bin index (binIdx) in the binarized representation of the syntax element, or adaptively chosen from two or more probability models depending on the related side information (e.g., spatial neighbors as illustrated in Fig. 8.1, component, depth or size of CU/PU/TU, or position within TU). Selection of the probability model is referred to as context

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00224

modeling. As an important design decision, the latter case is generally applied to the most frequently observed bins only, whereas the other, usually less frequently observed bins, will be treated using a joint, typically zero-order probability model. In this way, CABAC enables selective adaptive probability modeling on a sub-symbol level, and hence, provides an efficient instrument for exploiting inter-symbol redundancies at significantly reduced overall modeling or learning costs. Note that for both the fixed and the adaptive case, in principle, a switch from one probability model to another can occur between any two consecutive regular coded bins. In general, the design of context models in CABAC reflects the aim to find a good compromise between the conflicting objectives of avoiding unnecessary modeling-cost overhead and exploiting the statistical dependencies to a large extent.

The parameters of probability models in CABAC are adaptive, which means that an adaptation of the model probabilities to the statistical variations of the source of bins is performed on a bin-by-bin basis in a backward-adaptive and synchronized fashion both in the encoder and decoder; this process is called *probability estimation*. For that purpose, each probability model in CABAC can take one out of 126 different states with associated model probability values p ranging in the interval $[0.01875, 0.98125]$. The two parameters of each probability model are stored as 7-bit entries in a context memory: 6 bits for each of the 63 probability states representing the model probability p_{LPS} of the least probable symbol (LPS) and 1 bit for ν_{MPS}, the value of the most probable symbol (MPS). The probability estimator in CABAC is based on a model of "exponential aging" with the following recursive probability update after coding a bin b at time instance t:

$$p_{\mathrm{LPS}}^{(t+1)} = \begin{cases} \alpha * p_{\mathrm{LPS}}^{(t)}, & \text{if } b = \nu_{\mathrm{MPS}}, \text{ i.e., an MPS occurs} \\ 1 - \alpha * (1 - p_{\mathrm{LPS}}^{(t)}), & \text{otherwise.} \end{cases} \tag{8.1}$$

Here, the choice of the scaling factor α determines the speed of adaptation: A value of α close to 1 results in a slow adaptation ("steady-state behavior"), while faster adaptation can be achieved for the non-stationary case with decreasing α. Note that this estimation is equivalent to using a sliding window technique [4, 65] with window size $W_\alpha = (1 - \alpha)^{-1}$. In the design of CABAC, Eq. (8.1) has been used together with the choice of

$$\alpha = \left(\frac{0.01875}{0.5} \right)^{\frac{1}{63}} \quad \text{with} \quad \min_t p_{\mathrm{LPS}}^{(t)} = 0.01875, \tag{8.2}$$

and a suitable quantization of the underlying LPS-related model probabilities into 63 different states, to derive a finite-state machine (FSM) with tabulated transition rules [46]. This table-based probability estimation method was unchanged in HEVC, although some proposals for alternative probability estimators [1, 78] have shown average bit rate savings of 0.8–0.9 %, albeit at higher computational costs.

Each probability model in CABAC is addressed using a unique context index (ctxIdx), either determined by a fixed assignment or computed by the context

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00225

derivation logic by which, in turn, the given context model is specified. A lot of effort has been spent during the HEVC standardization process to improve the model assignment and context derivation logic both in terms of throughput and coding efficiency. More details on the specific choice of context models for selected syntax elements in HEVC are given in Sect. 8.4–8.6.

8.2.3 *Multiplication-Free Binary Arithmetic Coding: The M Coder*

Binary arithmetic coding, or arithmetic coding in general, is based on the principle of recursive interval subdivision. An initially given interval represented by its lower bound (base) L and its width (range) R is subdivided into two disjoint subintervals: one interval of width

$$R_{\mathrm{LPS}} = p_{\mathrm{LPS}} * R, \tag{8.3}$$

which is associated with the LPS, and the dual interval of width $R_{\mathrm{MPS}} = R - R_{\mathrm{LPS}}$, which is assigned to the MPS. Depending on the binary value to encode, either identified as LPS or MPS, the corresponding subinterval is then chosen as the new coding interval. By recursively applying this interval-subdivision scheme to each bin b_j of a given sequence $\mathbf{b} = (b_1, b_2, \ldots, b_N)$ of bins, the encoder finally determines a value $c_{\mathbf{b}}$ in the subinterval $[L^{(N)}, L^{(N)} + R^{(N)})$ that results after the Nth interval subdivision process. The (minimal) binary representation of $c_{\mathbf{b}}$ is the arithmetic code of the input bin sequence $\mathbf{b}$. To ensure that finite-precision registers are sufficient to represent $R^{(j)}$ and $L^{(j)}$ for all $j \in \{1, 2, \ldots, N\}$, a renormalization operation is required, whenever $R^{(j)}$ falls below a certain limit after one or more interval subdivision process(es). By renormalizing $R^{(j)}$, and accordingly $L^{(j)}$, the leading bits of the arithmetic code can be output as soon as they are unambiguously identified.

On the decoder side, the sequence of encoded binary values can be easily recovered by tracking the interval subdivision, including renormalization, according to Eq. (8.3) step-by-step and by comparing the bounds of both subintervals to the transmitted value representing the final subinterval. Note that the width $R^{(N)}$ of the final subinterval is proportional to the product $\prod_{j=1}^{N} p(b_j)$ of the individual model probability $p(b_j)$ assigned to the bins b_j of the bin sequence, such that for signaling the final subinterval, the lower bound of the empirical entropy of the bin sequence given by $-\log_2 \prod_{j=1}^{N} p(b_j) = -\sum_{j=1}^{N} \log_2 p(b_j)$ is approximately achieved.

From a practical implementation point of view, the most costly operation involved in binary arithmetic coding is given by the multiplication in Eq. (8.3). Even worse, if probability estimation is based on a simple scaled-count estimator using scaled cumulative frequency counts of bins, this operation may even involve an integer division. A solution to this problem was already proposed during the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00226

H.264/AVC standardization process by using a design of a family of multiplication-free binary arithmetic coders, which later became known as the *modulo coder* (M coder) [43, 45]. The main innovative features of this design are given by a table-based interval subdivision coupled with the above-mentioned FSM-based probability estimation as well as a fast bypass coding mode. The former, which is also the basis of what is called the *regular coding mode* of the M coder, will be briefly reviewed next, followed by a short discussion of the latter aspect.

8.2.3.1 Regular Coding Mode

The basic idea of the M-coder approach of interval subdivision is to quantize the range of possible interval widths induced by renormalization into a small number of K cells. To further simplify matters, a uniform quantization with $K = 2^\kappa$ is assumed to be performed, resulting in a set $\mathbf{W} = \{W_0, W_1, \cdots, W_{K-1}\}$ of representative interval widths. Together with the representative set of LPS-related probability values of the FSM given by $\mathbf{P} = \{p_0, p_1, \cdots, p_{N-1}\}$, this quantization enables the approximation of the multiplication on the right-hand side of Eq. (8.3) by means of a table of $K \times N$ pre-calculated product values $\{W_k * p_n \,|\, 0 \leq k < K;\, 0 \leq n < N\}$ in a suitable chosen integer precision. The entries of the corresponding 2-D lookup table *TabRangeLPS* are addressed by the (probability) state index n and the quantization cell index $k(R)$ related to the given value of the interval range R. Computation of $k(R)$ is easily carried out by a concatenation of a bit shift and a bit-masking operation, where the latter can be interpreted as a *modulo operation* using the operand $K = 2^\kappa$, hence the naming of the family of coders.

In the context of H.264/AVC, the optimal empirical choice of the free parameters $\kappa = 2$ and $N = 64$ was determined under the constraint of a maximum table size of $2^\kappa * N \leq 256$ bytes for the lookup table *TabRangeLPS* with each of its entries being represented with 8 bits. This specific M-coder design of using a lookup table *TabRangeLPS* with 4×64 entries was also adopted for HEVC. Please note that by choosing a value of $\kappa = 0$, the 2-D table *TabRangeLPS* degenerates to a 1-D table, where for all possible values of R only one single representative interval width value W is used for each of the N product values $p_n * R$, where $0 \leq n < N$. This choice is equivalent to the subinterval division operation performed in the Q coder and its derivatives of QM and MQ coder, as has been standardized in JBIG, JPEG, and JPEG2000. Thus, the M-coder design can be interpreted as a generalization of the Q-coder family.[2] Compared to the QM/MQ coder, the M coder, being configured as in H.264/AVC and HEVC, achieves an increase in throughput of 18 %, while at the same time it provides bit-rate savings of 2–4 %, when evaluated in the CABAC environment of H.264/AVC [43]. Interestingly, the throughput improvements of

[2]Please note that apart from the interval subdivision aspect there are some subtle technical differences between (and also within) the coder families, such as concerning, e.g., probability estimation, conditional exchange, carry-over handling, and termination.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00227

the M coder can be largely attributed to its unique bypass functionality, as being reviewed in the next subsection, while its use of a larger lookup table for interval subdivision generates the main effects in coding-efficiency gain; however, this increased table size can also adversely affect the overall throughput gain of the M coder.

8.2.3.2 Bypass Coding Mode

As already mentioned, most of the throughput improvements of the M coder relative to the Q-coder technology can be attributed to its second innovative feature, which is given by a bypass of the probability estimation for approximately uniform distributed bins. In addition, the interval subdivision is substituted by a hard-wired equipartition in this so-called bypass coding mode. In this way, the whole encoding/decoding process (including renormalization) can be realized by nothing more than a bit shift, a comparison, and for half of the symbols an additional subtraction.

Bypass coding has become an even more important feature during the HEVC standardization process. While in H.264/AVC bypass coding was mainly used for signs and least significant bins of absolute values of quantized transform coefficients, in HEVC the majority of possible bin values is handled through the bypass coding mode. As noted above, this is also a consequence of carefully designed binarization schemes, which already serve as a kind of near-optimal prefix-free codes of the corresponding syntax elements.

8.2.3.3 Fast Renormalization

One of the major throughput bottlenecks in any arithmetic encoding and decoding process is given by the renormalization procedure. Renormalization in the M coder is required whenever the new interval range R after interval subdivision no longer stays within its admissible domain. Each time a renormalization operation must be carried out, one or more bits can be outputted at the encoder or, equivalently, have to be read by the decoder. This process, as it is specified in H.264/AVC and HEVC, is performed bit-by-bit and is controlled by some conditional branches to check each time if further renormalization loops are required. Both conditional branching and bitwise processing, however, constitute considerable obstacles to a sufficiently high throughput.

As a mitigation of this problem, a fast renormalization policy for the M coder was proposed in [48]. By replacing the conventionally bitwise performed operations in the regular coding mode with byte-wise or word-wise processing, a considerably increased decoder throughput of around 25 % can be achieved. The corresponding non-normative, fully standard-compliant changes were integrated into the reference software implementations of both H.264/AVC and HEVC. For more details, please refer to [47, 48].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00228

8.2.3.4 Termination

For termination of the arithmetic codeword in the M coder a special, non-adapting probability state is reserved. The corresponding probability state index is given by $n = 63$ and the corresponding entries of *TabRangeLPS* deliver a constant value of $R_{\mathrm{LPS}} = 2$. As a consequence, for each terminating syntax element, such as `end_of_slice_segment_flag`, `end_of_sub_stream_one_bit`, or `pcm_flag`, 7 bits of output are generated in the renormalization process. Two more bits are needed to be flushed in order to properly terminate the arithmetic codeword. Note that the least significant bit in this flushing procedure, i.e., the last written bit at the encoder, is always equal to 1 and thus, represents the so-called `rbsp_stop_one_bit`. Before packaging of the bitstream, the arithmetic codeword is filled up for byte alignment with zero-valued alignment bits.

8.3 Design Considerations

Most of the proposals submitted to the joint Call for Proposals on HEVC in April 2010 already included some form of advanced entropy coding. Some of those techniques were based on improved versions of CAVLC or CABAC, others were using alternative methods of statistical coding, such as V2V (variable-to-variable) codes [31] or PIPE (probability interval partitioning entropy) codes [50, 51, 102], and a third category introduced increased capabilities for parallel processing on a bin level [84], syntax element level [94, 96], or slice level [25, 28, 105]. In addition, improved techniques for coding of transform coefficients, such as zero-tree representations [2], alternate scanning schemes [40], or template-based context models [55, 102], were proposed.

After an initial testing phase of video coding technology from the best performing HEVC proposals, it was decided to start the first HEVC test model (HM1.0) [53] with two alternate configurations similar to what was given for entropy coding in H.264/AVC: a high efficiency configuration based on CABAC and a low-complexity configuration based on LCEC as a CAVLC surrogate. Interestingly enough, the CABAC-based entropy coding of HM1.0 already included techniques for improving both coding efficiency and throughput relative to its H.264/AVC-related predecessor. To be more specific, a template-based context modeling scheme for larger transform block sizes [49, 55] and a parallel context processing technique for selected syntax elements of transform coefficient coding [10] were already part of HM1.0. During the subsequent collaborative HEVC standardization phase, more techniques covering both aspects of coding efficiency and throughput were integrated, as will be discussed in more details in the following.

While CABAC inherently is targeting at high coding efficiency, its data dependencies can cause it to be a throughput bottleneck, especially at high bit rates as was already analyzed in the context of H.264/AVC [95]. This means that, without any further provision, it might have been difficult to support the growing throughput

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00229

requirements for future video codecs. Furthermore, since high throughput can be traded-off for power savings using voltage scaling [14], the serial nature of CABAC may limit the battery life for video codecs that reside on mobile devices. This limitation is a critical concern, as a significant portion of video codecs today are running on battery-operated devices. Accordingly, both coding efficiency and throughput improvement tools as well as the trade-off between these two requirements were investigated in the standardization of entropy coding for HEVC. The trade-off between coding efficiency and throughput comes from the fact that, in general, dependencies are a result of removing redundancy which, in turn, improves coding efficiency; however, increasing dependencies usually makes parallel processing more difficult which, as a consequence, may degrade throughput. This section describes the various techniques used to improve both coding efficiency and throughput of CABAC entropy coding for HEVC.

8.3.1 Brief Summary of HEVC Block Structures and CABAC Coding Efficiency Improvements

In the evolutionary process from H.264/AVC to HEVC, improved coding efficiency for CABAC entropy coding was addressed in a number of proposals, such as [24, 102, 106]. The majority of coding-efficiency related CABAC proposals in the HEVC standardization process was oriented towards transform coefficient coding, since at medium to high bit rates the dominant part of bits is consumed by syntax elements related to residual coding. As a consequence, this subsection will focus on considerations that were made with regards to the specific CABAC design for those syntax elements. Note, however, that due to the more consistent design of HEVC in terms of tree structures for both partitioning of prediction blocks and transform blocks, special care has also been taken to ensure an efficient modeling and coding of the corresponding tree structuring elements. In addition, for new coding tools in HEVC, such as block merging and sample adaptive offset (SAO) in-loop filtering, additional assignments of binarization and context modeling schemes were needed.

Transform coding in HEVC is based on a tree-structured variable block size approach with the corresponding quadtree structure referred to as *residual quadtree* (RQT) [49, 102]. RQTs are nested into the leaves of another quadtree, the so-called *coding quadtree* (CQT), which determines the subdivision of each block of $2^N \times 2^N$ luma samples, referred to as a coding tree block (CTB) [49, 102]. The block partitioning for both prediction and transform coding is the same for luma and chroma picture component samples,[3] and hence, a common coding and residual quadtree syntax is used to signal the partitioning. As a result, the blocks of luma and chroma samples and associated syntax elements are grouped together in a so-called *unit*.

[3]There is one exception to this general rule in HEVC, which is discussed in more detail in Chap. 3.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00230

A transform unit (TU) aggregates the transform blocks (TBs) of luma and chroma samples as well as the syntax elements used to represent the associated transform coefficient levels. Each TU and the related luma and two chroma TBs are determined as a leaf of the corresponding RQT. Supported TB sizes for both luma and chroma are in the range from 4×4 to 32×32 samples, where the corresponding core transforms are separable applications of a fixed-point approximation of the 1-D Discrete Cosine Transform (DCT) for dyadically increasing lengths from 4 to 32 points [26]. An exception is given for 4×4 luma TBs of residual signals resulting from intra-picture predicted blocks, where instead of the DCT-like core transform a separable fixed-point approximation of the 1-D Discrete Sine Transform (DST) is used [100].

Note that a prediction unit (PU) aggregates the prediction blocks (PBs) of luma and chroma samples and the associated syntax elements like motion data. A coding unit (CU) encapsulates the luma and chroma coding block (CB) samples and the so-called prediction mode, i.e., the decision whether the corresponding samples are coded using intra-picture or inter-picture prediction, as well as some additional syntax elements. On the top level of the hierarchy, a coding tree unit (CTU) comprises the CTBs of luma and chroma samples, the associated CQT syntax structure and all CUs at the CQT leaves.

8.3.1.1 Coefficient Grouping into Subblocks

Given the larger variety of TB sizes, one of the primary goals of CABAC entropy coding for transform coefficient data in HEVC was to achieve a design that uses for all block sizes as much of the same logic and the same procedures as possible. Although at first glance this objective seems to be somehow unrelated to coding efficiency, it turns out that at least one particular element leading to such a unified design is also crucial for achieving high coding efficiency. This coding element is given by the grouping of coefficients into so-called subblocks of size 4×4 for transform blocks with size greater than 4×4. Subblocks were first proposed in [49, 55, 102] and became part of HM1.0. In the subsequent HEVC development process, their use was iteratively refined and extended in a way as will be explained in more detail in Sect. 8.6.

8.3.1.2 Hierarchy of Significance Flags

Since for most common coding conditions, a large portion of transform coefficients is quantized to zero, or equivalently, the representation of the residual signal in the DCT-/DST-like basis functions is supposed to be sparse, a hierarchical structured set

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00231

of four different significance flags[4] is introduced in HEVC to reduce the number of individual significance flags to be transmitted. This hierarchy of syntax elements also reflects the hierarchical processing of TBs within the RQT as well as the processing of subblocks within a given TB.

The use of so-called *coded block flags* (CBF), indicating the occurrence of significant, i.e., nonzero transform coefficients in a TB, was already part of H.264/AVC CABAC-based residual coding. In HEVC, this concept was extended to also cover the RQT root on the top level of the hierarchy as well as the subblock on a lower level of the hierarchy. Consequently, there are a `rqt_root_cbf`, at least for RQT roots in inter-predicted CUs, `cbf_luma`, `cbf_cb`, and `cbf_cr` for the visited TBs of the three color components, and a `coded_sub_block_flag` (CSBF) for each visited subblock in a TB. On the lowest level of the hierarchy, for each visited subblock a so-called *significance map* indicates the location of nonzero coefficients for each scan position in a subblock.

This hierarchy of significance flags is complemented by the syntax elements indicating the last significant scan position in a TB, which somehow serve as an entry point into each significant TB and which is equivalent to signaling the insignificance of a partial area of a TB. The latter concept differs from H.264/AVC, where for each `significant_coeff_flag` (SIG) with a value of one, a `last_significant_coefficient_flag` (LAST) is signaled indicating if the current scan position is the last nonzero coefficient inside the TB. Note that this latter signaling scheme is equivalent to using a TrU binarization (with inverted bin values) for the number of nonzero coefficients in a TB, such that each bin of the resulting bin sequence is intertwined with the corresponding nonzero significance flag. This design aspect of mixing two flags on a bin level in H.264/AVC was later found to be critical in terms of throughput, as will be discussed in Sect. 8.6.

8.3.1.3 Context Modeling for Coding of Significance Flags

Particular care has been taken to properly specify the context models for coding of significance flags. For instance, modeling of the CBF is based on the RQT depth, while that for the CSBF is using neighboring CSBF information. For coding of the significance map, which typically consumes most of the bits in HEVC transform coding, additional dependencies between neighboring elements have been exploited, at least for TBs larger than 4×4. Initially, for that purpose a local template was proposed [49, 55, 102] and adopted for HM1.0. Although this design provides high coding efficiency, it introduces some critical data dependencies. As a solution to this problem, a combination of position-based information (as used in H.264/AVC) and template-based neighborhood information was finally adopted for context modeling of significance map entries [41, 77]. This particular

[4]Note that the term "significance flag" is interpreted here and in the following in a much broader sense than originally used in the context of H.264/AVC.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00232

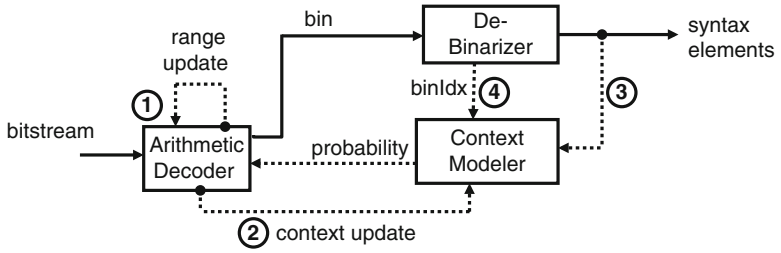


Fig. 8.2 Three key operations in CABAC (from a decoder perspective): Binarization, Context Modeling/Selection and (Binary) Arithmetic Coding. Feedback loops in the decoder are highlighted with *dashed lines*

example also illustrates how both aspects of coding efficiency and throughput were considered during the HEVC standardization process in a balanced way. More on the throughput aspects is given in the next subsection, while the details of context modeling for all syntax elements related to residual coding are provided in Sect. 8.6.

8.3.2 CABAC Throughput Bottlenecks

CABAC, as originally designed for H.264/AVC and also, as initially selected for the HEVC standardization starting point in HM1.0, has some serious throughput issues (particularly for decoder implementations at higher bit rates) [80, 95]. The throughput of CABAC is determined based on the number of binary symbols (bins) that it can process per second. The throughput can be improved by increasing the number of bins that can be processed in a cycle. However, the data dependencies in CABAC make processing multiple bins in parallel difficult and costly to achieve. These dependencies result in feedback loops in the CABAC decoder as shown in Fig. 8.2, and can be described as follows:

1. The updated range is fed back for recursive interval subdivision.
2. The updated context is fed back for probability estimation.
3. The context modeler selects the probability model based on the type of syntax element and, as already noted above, for selected syntax elements, based on some derivation process that involves other previously decoded bin values or other relevant side information. At the decoder, for non-binary syntax elements, the decoded bin value is fed back to determine whether to continue processing the same syntax element or to switch to another syntax element. If a switch occurs, the value of the decoded bin may also be used to determine which syntax element to decode next.
4. The context modeler may also select the probability model based on the bin position in the syntax element (binIdx). At the decoder, the decoded bin value is fed back to determine whether to increment binIdx and continue to decode the current syntax element, or set binIdx equal to 0 and switch to another syntax element.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00233

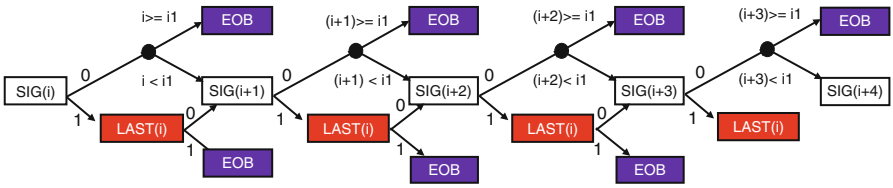


Fig. 8.3 Context speculation required to achieve 5× parallelism when processing the significance map in H.264/AVC. Notation: i = coefficient position; i1 = MaxNumCoeff (BlockType)−1; EOB = end of block; SIG = `significant_coeff_flag`; LAST = `last_significant_coeff_flag`

Note that the feedback loops have different degrees of impact on throughput. The range update (1) and context update (2) feedback loops are simpler than the context modeling loops (3, 4) and thus do not affect throughput as severely. If the context of a bin depends on the value of another bin being decoded in parallel, then speculative computations are required, which increases area cost and critical path delay [94]. The amount of speculation can grow exponentially with the number of parallel bins, which limits the throughput that can be achieved [80]. Figure 8.3 shows an example of the speculation tree for significance map in H.264/AVC. Thus the throughput bottleneck is primarily due to the context modeling dependencies.

8.3.3 Summary of Techniques for CABAC Throughput Improvements

Several techniques were used to improve the throughput of CABAC in HEVC [88]. There was a lot of effort spent in determining how to use these techniques with minimal coding loss. They were applied to various parts of entropy coding in HEVC and will be referred to throughout the rest of this chapter.

8.3.3.1 Reduce Regular Coded Bins

The throughput is limited for regular coded bins due to the data dependencies described in Sect. 8.3.2. However, it is easier to process bypass coded bins in parallel since they do not have the data dependencies related to context modeling (i.e., feedback loops 2, 3 and 4 in Fig. 8.2). In addition, arithmetic coding for bypass bins is simpler as it only requires a right shift versus a table look up for regular coded bins. Thus, the throughput can be improved by reducing the number of regular coded bins and using bypass coded bins instead [16, 54, 58, 59].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00234

8.3.3.2 Group Bypass Coded Bins

Multiple bypass bins can be processed in the same cycle only if they occur *consecutively* within the bitstream. Thus, bins should be reordered such that bypass coded bins are grouped together in order to increase the likelihood that multiple bins are processed per cycle [19, 67, 87].

8.3.3.3 Group Bins with Same Context

Processing multiple regular coded bins in the same cycle often requires speculative calculations for context modeling. The amount of speculative computations increases if bins using different contexts and context modeling logic are interleaved, since numerous combinations and permutations must be accounted for. Thus, to reduce speculative computations, bins should be reordered such that bins with the same contexts and context modeling logic are grouped together so that they are likely to be processed in the same cycle [9, 10, 73]. This also reduces context switching resulting in fewer memory accesses, which also increases throughput and reduces power consumption. This technique was first introduced in [10] and was referred to as parallel context processing (PCP) throughout the standardization process.

8.3.3.4 Reduce Context Modeling Dependencies

Speculative computations are required for multiple bins per cycle decoding due to the dependencies in the context modeling. For instance, this is an issue when the context modeling for the next bin depends on the decoded value of the current bin. Reducing these dependencies simplifies the context modeling logic and reduces the amount of speculative calculations required to process multiple bins in parallel [18, 80, 85].

8.3.3.5 Reduce Total Number of Bins

In addition to increasing the throughput, it is desirable to reduce the workload itself by reducing the total number of bins that need to be processed. This can be achieved by changing the binarization, inferring the value of some bins,[5] and sending higher level flags to avoid signaling redundant bins [12, 56, 59].

[5]The benefit of inferring bins must be traded-off with a potential increase in context selection complexity.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00235

8.3.3.6 Reduce Parsing Dependencies

As parsing with CABAC may constitute a throughput bottleneck, it is important to minimize any dependency on other video decoding processes, which could cause CABAC to stall or may even prevent a successful parsing process in case of picture loss due to transmission errors [7, 79, 108] (see Sect. 8.5.1.1). Ideally the parsing process should be decoupled from all other decoding processes, which actually is the case for CABAC in H.264/AVC. Decoupling parsing from the sample reconstruction process is also important when entropy decoupling is used, i.e., when a large frame level buffer is inserted between the entropy decoder and the rest of the decoder to absorb the variance in the bit-rate and pixel-rate workloads, respectively.

8.3.3.7 Reduce Memory Requirements

Memory accesses often contribute to the critical path delay. Thus, reducing memory storage requirements is desirable as fewer memory accesses increases throughput as well as reduces implementation cost and power consumption [81, 90].

8.4 Coding Tree Unit and Coding Unit Syntax Elements

In HEVC, a picture is partitioned into a regular grid of disjoint square blocks of $2^N \times 2^N$ luma samples and, in case of 4:2:0 color sampling, corresponding square blocks of $2^{N-1} \times 2^{N-1}$ chroma samples. The parameter $N = 4, 5$, or 6 can be chosen by the encoder and transmitted in the sequence parameter set (SPS), such that the corresponding coding tree units represent luma CTBs of size 16×16, 32×32, or 64×64 samples, respectively. The CTU syntax elements describe how the corresponding CTBs can be further partitioned into smaller coding blocks by use of the coding quadtree and how the method of sample adaptive offset (SAO) in-loop filtering is performed on the reconstructed luma and chroma samples belonging to the CTU.

Within a picture, an integer number of CTUs can be grouped into a slice. Each slice itself consists of one (leading) independent slice segment and zero or more subsequently ordered dependent slice segments. A flag called `end_of_slice_segment_flag` is sent to indicate the last CTU in a slice segment. In addition, tiles and wavefront parallel processing, which are introduced in Chap. 3, can be used to fragment the slice segment into multiple substreams,[6] each being represented by its own CABAC codeword. Therefore, if `end_of_slice_segment_flag` indicates that it is not the last CTU in a slice segment, a flag called `end_of_sub_stream_one_bit` is used to indicate whether it is the

[6] Slice segments can also be used to fragment tiles and wavefronts into substreams.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00236

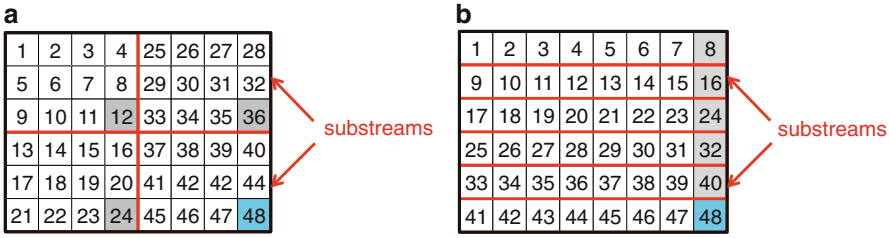


Fig. 8.4 These two examples illustrate which CTUs are terminated when slice segments are divided into substreams using tiles and wavefront parallel processing. Values of (`end_of_slice_segment_flag`, `end_of_sub_stream_one_bit`) are given for each configuration. (**a**) *Tiles*: CTU 12, 24, and 36 have (0, 1); CTU 48 (1, not signaled); and the rest of the CTUs have (0,0). (**b**) *Wavefront parallel processing*: CTU 8, 16, 24, 32 and 40 have (0, 1); CTU 48 (1, not signaled); and the rest of the CTUs have (0,0)

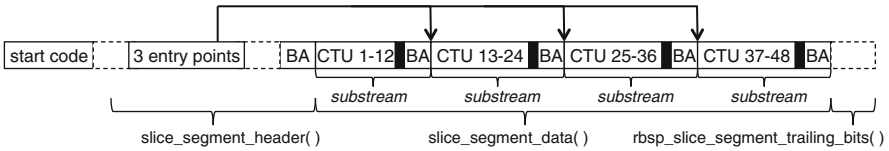


Fig. 8.5 Ordering of the bitstream for the tiles example in Fig. 8.4a. CABAC needs to be terminated before byte alignment (*BA*) as shown by the *black boxes*. Entry points for substreams are sent in *slice_segment_header()*

last CTU of the corresponding substream.[7] An example of this is illustrated in Fig. 8.4. Both `end_of_slice_segment_flag` and `end_of_sub_stream_one_bit` are coded using the terminating mode of the arithmetic coding engine. This is required since at the end of a slice segment or a substream, the arithmetic coding engine must be flushed and the resulting CABAC codeword must be byte aligned before, at least in the former case, inserting the startcode for the next slice or entry point for the next slice segment. Figure 8.5 shows an example of the locations of CABAC termination within a bitstream.

8.4.1 Coding Block Structure

The coding block structure is determined by the coding quadtree which is signaled by a flag called `split_cu_flag` at each of its nodes to indicate whether a given coding block should be further subdivided into four smaller CBs. There is a strong spatial correlation between the chosen CQT depth of neighboring CBs, i.e., the

[7]Note that the value of `end_of_sub_stream_one_bit` is always 1 and it is only sent for the last CTU of a substream.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00237

block sizes of neighboring CBs, thus the context selection for `split_cu_flag` depends on the relative depth of the top and left neighboring CBs compared to that of the current CB. Note that in H.264/AVC the partitioning information is sent together with other data as aggregated syntax elements `mb_type` and `sub_mb_type` with different ranges of allowed values and hence different binarization schemes for different slices.[8] This kind of aggregating different information in one single syntax element is mostly due to historical reasons, reflecting the circumstances that earlier video coding standards (including H.264/AVC) were designed under the regime of VLC-based entropy coding, where alphabet extensions are used to circumvent the lower bound of 1 bit per symbol. Thus, by allowing the signaling of a coding quadtree structure with a one-bin syntax element, i.e., the `split_cu_flag` at each node, HEVC is much more flexible and allows many more coding and prediction block structures than H.264/AVC, even when choosing a CTB size of 16×16 luma samples and ignoring the fact that HEVC doesn't allow for inter-predicted 4×4 luma blocks, as discussed in Chap. 3.

8.4.2 Prediction Mode and Prediction Block Structure

In P and B slices, a `cu_skip_flag` is sent for each CU to indicate whether all associated CBs are coded using skip mode, i.e., by using the so-called merge mode for inter-picture prediction (as explicitly described in Sect. 8.5) and not sending any residual data. To leverage spatial correlation of neighboring CUs, the context of the `cu_skip_flag` depends on whether the top and left neighboring CUs are also skipped. For every non-skipped CU, a regular coded flag called `pred_mode_flag` is sent to indicate the prediction mode, i.e., the decision whether the CU is either intra coded or inter coded.[9]

Every non-skipped CU may be further subdivided into PUs, as shown for the example in Fig. 8.6. The syntax element `part_mode` indicates if and how each CU is partitioned for the purpose of prediction. The choice of prediction block structures for each CU depends on whether the CU is intra-coded or inter-coded; accordingly, `part_mode` is binarized and coded differently for intra-coded CUs and inter-coded CUs, as shown in Fig. 8.7.

Intra-coded CUs can have a single PU (referred to as PART_2Nx2N) equal to the size of the CU, or be subdivided into four smaller PUs (referred to as PART_NxN). PART_NxN, however, is only allowed when the CU size is equal to the minimum allowed CU size. If the CU size is greater than the minimum allowed CU size,

[8] In H.264/AVC, `mb_type` and `sub_mb_type` are used to represent the following equivalent information in HEVC: `split_cu_flag`, `pred_mode_flag`, `part_mode`, `pcm_flag`, `inter_pred_idc`, coded block pattern (`cbf_luma`, `cbf_cr`, `cbf_cb`) and intra prediction mode for 16×16 intra-coded PU.

[9] `pred_mode_flag` is not sent for CUs in I slices since they are all intra-coded.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00238

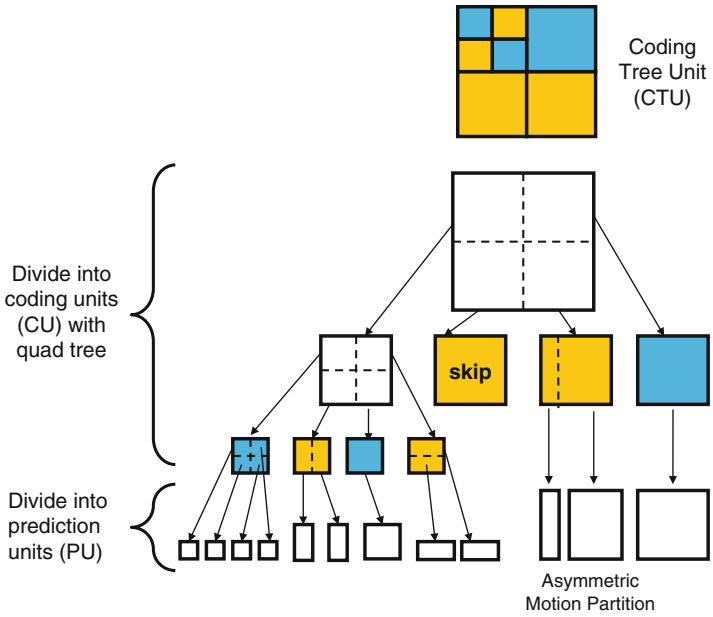


Fig. 8.6 A coding tree unit is subdivided into CUs along the associated coding quadtree. Each resulting CU may be further subdivided into PUs. The intra-coded CUs are in *blue* while inter-coded CUs are in *orange*. Note that the figure only shows the corresponding CTB, CBs, and PBs of the luma component

`split_cu_flag` is used instead of `part_mode` to avoid redundant signaling. For instance, if the minimum CU size is 8×8 in terms of luma samples, a CU of size 16×16 with four 8×8 PUs is signaled with `split_cu_flag`= 1, and `part_mode`=PART_2Nx2N rather than `split_cu_flag`= 0 and `part_mode`=PART_NxN. Accordingly, `part_mode` is not signaled but inferred to be PART_2Nx2N when the CU size is greater than the minimum allowed CU size. If the CU size is equal to the minimum allowed CU size, `part_mode` is coded using a flag with a fixed context for a given slice type.

Inter-coded CUs have more prediction partitioning options than intra-coded CUs. In addition to PART_2Nx2N and PART_NxN, inter-coded CU can also be subdivided into two rectangular PUs in either horizontal (PART_Nx2N) or vertical directions (PART_2NxN). In case of enabling the so-called asymmetric motion partitioning (AMP), the additional prediction partitioning possibilities PART_2NxnU, PART_2NxnD, PART_nLx2N, and PART_nRx2N are supported. Custom binarization is used for `part_mode` as shown in Fig. 8.7b. The first bin indicates whether or not the CU is partitioned into smaller PUs. If the CU size is greater than the minimum allowed CU size, the second bin indicates the direction of the partition (vertical or horizontal), the third bin indicates whether AMP is used and if so, then a fourth bin is sent to indicate which asymmetric partition is used in

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00239

a

part_mode	CU Size > min CU Size	CU Size = min CU Size
PART_2Nx2N	Inferred	1
PART_NxN	Not Allowed	0

b

part_mode	CU Size > min CU Size: AMP disabled	CU Size > min CU Size: AMP enabled	CU Size = min CU Size: CU Size = 8x8	CU Size = min CU Size: CU Size > 8x8
PART_2Nx2N	1	1	1	1
PART_Nx2N	0 1	0 1 1	0 1	0 1
PART_2NxN	0 0	0 0 1	0 0	0 0 1
PART_NxN				0 0 0
PART_2NxnU	Partition into smaller PU	0 1 0 0	Partition into smaller PU	Partition into smaller PU
PART_2NxnD	Direction of PU	0 1 0 1	Direction of PU	Truncated unary for PU size
PART_nLx2N		0 0 0 0		
PART_nRx2N		0 0 0 1		

Select AMP
Use Asymmetric Partitions (AMP)
Direction of PU
Partition into smaller PU

Fig. 8.7 Context selection and binarization of part_mode. *Underlined symbols* are bypass coded. (**a**) Intra-coded CU, (**b**) inter-coded CU

the given direction. If the CU size is equal to the minimum allowed CU size, AMP is not allowed and truncated unary coding is used to indicate if the partitions are PART_Nx2N, PART_2NxN, and PART_NxN, respectively.[10] A different context is

[10]Note that the minimum allowed inter-coded PART_NxN size is 8 × 8, so for CU size equal to 8 × 8, the only allowed partitions are PART_2NxN and PART_Nx2N, and cMax of 2 is used for truncated unary.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00240

used for the first and second bin to estimate the probabilities of whether the PU is partitioned into smaller PUs, and the direction of the PU. Two different contexts are used for the third bin depending on when the CU size is greater or equal to the minimum allowed CU size. In the former, the context is based on the probability of whether asymmetric partitions are used, while in the latter, the context is based on the probability of whether PART_NxN is used. To reduce the number of regular coded bins, the fourth bin (for AMP) is bypass coded.

8.4.3 Signaling of Special Coding Modes

HEVC supports two special coding modes, which are invoked on a CU level: the so-called I_PCM mode and the lossless coding mode. Both modes, albeit similar in appearance to some degree, serve different purposes and hence, use different syntax elements for providing different functionalities.

A `pcm_flag` is sent to indicate whether all samples of the whole CU are coded with *pulse code modulation* (PCM), such that prediction, transform, quantization, and entropy coding as well as their counterparts on the decoder side are simply bypassed. This I_PCM mode, however, is only allowed for intra-coded CUs with prediction partitioning mode PART_2Nx2N.[11] The `pcm_flag` is coded with the termination mode of the arithmetic coding engine, since in most cases I_PCM mode is not used, and if it is used, the arithmetic coding engine must be flushed and the resulting CABAC codeword must be byte aligned before the PCM sample values can be written directly into the bitstream with fixed length codewords.[12] This procedure also indicates that the I_PCM mode is in particular useful in cases, where the statistics of the residual signal would be such that otherwise, an excessive amount of bits would be generated when applying the regular CABAC residual coding process.

The option of *lossless coding*, where for coding of the prediction residual both the transform and quantization (but not the entropy coding) are bypassed, is also enabled on a CU level and indicated by a regular coded flag called `cu_transquant_bypass_flag`. The resulting samples of the losslessly represented residual signal in the spatial domain are entropy coded by the CABAC residual coding process (see Sect. 8.6), as if they were conventional transform coefficient levels. Note that in lossless coding mode, both in-loop filters are also bypassed in the reconstruction process (which is not necessarily the case for I_PCM), such that a mathematically lossless (local) reconstruction of the input signal is achieved.

[11] I_PCM is not allowed for intra-coded 4×4 blocks.

[12] Note that the PCM sample bit depth (i.e., wordlength) for luma and chroma samples can be specified independently in the SPS.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00241

8.4.4 Signaling of Block-Based Quantization Parameter Change

In the regular, i.e., lossy residual coding process, a different quantizer step size can be used for each CU to improve bit allocation, rate control, or both. Rather than sending the absolute quantization parameter (QP), the difference in QP steps relative to the slice QP is sent in the form of a so-called delta QP. This functionality can be enabled in the picture parameter set (PPS) by using the syntax element `cu_qp_delta_enabled_flag`.

In H.264/AVC, `mb_qp_delta` is used to provide the same instrument of delta QP at the macroblock level. The value of `mb_qp_delta` can range from $-(26 + \text{QpBdOffset}_Y/2)$ to $25 + \text{QpBdOffset}_Y/2$. For 8-bit video, this is -26 to 25, while for 10-bit video this is -32 to 31. `mb_qp_delta` is unary coded and thus requires up to 53 bins for 8-bit video and 65 bins for 10-bit video. All bins are regular coded.

In HEVC, delta QP is represented by the two syntax elements `cu_qp_delta_abs` and `cu_qp_delta_sign_flag`, if `cu_qp_delta_enabled_flag` in the PPS indicates so. The sign is sent separately from the absolute value, which reduces the average number of bins by half [23]. `cu_qp_delta_sign_flag` is only sent if the absolute value is non-zero. The absolute value is binarized with TrU (cMax=5) as the prefix and EG0 as the suffix [98]. The prefix is regular coded and the suffix is bypass coded. The first bin of the prefix uses a different context than the other four bins in the prefix (which share the same context) to capture the probability of having a zero-valued delta QP. Note that syntax elements for delta QP are only signaled for CUs that have non-vanishing prediction errors (i.e., at least one non-zero transform coefficient). Conceptually, the delta QP is an element of the transform coding part of HEVC and hence, can also be interpreted as a syntax element that is always signaled at the root of the RQT, regardless which transform block partitioning is given by the RQT structure. Table 8.3 shows examples of how delta QP is signaled for H.264/AVC and HEVC.

8.4.5 Signaling of SAO Parameters

SAO is a form of in-loop filtering that was introduced in HEVC. It is used to process the output of samples from the deblocking filter process and is the last step of the decoding process. SAO involves sample based processing rather than block based processing. There are two types of filtering: edge offset and band offset.

Edge offset (EO) involves comparing the sample and its neighboring sample values in one of four angular directions (horizontal, vertical, 45°, 135°).[13] The sample is compared to its neighbors in the selected direction (e.g., the sample has a lower value than both its neighbors); based on the comparison, the sample is

[13]Direction is also referred to as class in the HEVC specification.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00242

Table 8.3 Coding of delta QP in HEVC and H.264/AVC

Value	HEVC		H.264/AVC
	cu_qp_delta_abs	cu_qp_delta_sign_flag	mb_qp_delta
0	0	n/a	0
1	10	0	10
-1	10	1	110
2	110	0	1110
-2	110	1	1111 0
3	1110	0	1111 10
-3	1110	1	1111 110
4	11110	0	1111 1110
-4	11110	1	1111 1111 0
5	11111 0	0	1111 1111 10
-5	11111 0	1	1111 1111 110
6	11111 100	0	1111 1111 1110
-6	11111 100	1	1111 1111 11110
25	11111 111100101	0	1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 10
-26	11111 111100110	1	1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 0

Underlined symbols are bypass coded

assigned to a category, which determines the offset that is added to the sample. The value of the offset for a given category is set by the encoder. Band offset (BO) involves dividing the intensity range into four bands and then adding a different offset to the sample depending which band its sample intensity belongs to. For more details on SAO, please refer to Chap. 7.

The type, direction and offsets used to define the SAO filter can change for each CTB; however, all samples belonging to a CTB are processed with the same SAO filter (but luma and chroma CTBs may use different SAO filters). The SAO type is signaled using `sao_type_idx_luma` and `sao_type_idx_chroma` with TrU binarization. The first bin indicates whether the SAO filter is enabled and is regular coded, while the second bin indicates if edge or band offset is used and is bypass coded.

If edge offset is used, the direction of the edge is signaled using `sao_eo_class_luma` and `sao_eo_class_chroma` with FL binarization of two bins, all of which are bypass coded. If band offset is used, four `sao_band_position` syntax elements are signaled to indicate the start position of each band with a FL binarization of five bins, all of which are bypass coded.

For both types of SAO filtering, four `sao_offset_abs` are signaled (one for each category or band) using TrU with cMax computed by Eq. (8.4) and all bins are bypass coded.

$$\text{cMax} = (1 << (\min(\text{bitDepth}, 10) - 5)) - 1 \tag{8.4}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00243

Table 8.4 Differences in signaling between CTU/CU layer in HEVC and MB layer in H.264/AVC

	HEVC	H.264/AVC
Prediction and Coding Block Structure and Prediction Mode	`cu_skip_flag`, `split_cu_flag`, `pred_mode_flag`, `part_mode`	`mb_skip_flag`, `mb_type`, `sub_mb_type`
Maximum number of bins for delta QP	5 (regular), 10 (bypass)	53 (regular)
Maximum number of bins for SAO parameters	4 (regular), 113 (bypass)	n/a

For the band offset, the `sao_offset_sign` is signaled only when the offset is non-zero to reduce the total number of bins [36], while for edge offset the sign is inferred from the category [40].

To leverage the spatial correlation across CTBs, `sao_merge_left_flag` and `sao_merge_up_flag` are used to indicate if SAO parameters can be inherited from neighboring CTBs, which reduces signaling overhead. Both of these flags are regular coded using separate context models.

Significant effort was made to reduce the number of regular coded bins required to represent SAO filter syntax elements. As a result, the only regular coded bins are the merge flags and the first bin of the `sao_type_idx_luma` and `sao_type_idx_chroma` with the latter indicating whether SAO is enabled for luma and chroma CTBs, respectively.

8.4.6 Comparison of HEVC and H.264/AVC

Table 8.4 highlights the differences in signaling between the CTU/CU layer in HEVC and the macroblock (MB) layer in H.264/AVC, when processing 8-bit video. For a comparable block partitioning, HEVC typically produces fewer regular coded bins than H.264/AVC. At the same time, some of those regular coded bins in addition to those of the skip flag are adaptively selected based on CU depth, size and neighbors in HEVC, which improves coding efficiency relative to H.264/AVC. In general, however, the total amount of bits spent for signaling at the CTU/CU or MB layer is lower by more than an order of magnitude compared to the total amount of bits spent for transform coefficient level coding. As already discussed above and summarized in Table 8.4, the majority of bypass bins for the SAO parameters are due to the signaling of the offsets, while for H.264/AVC an excessive number of bins is only generated in the rare cases where large delta QP values have to be transmitted.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00244

8.5 Prediction Unit Syntax Elements

The prediction unit (PU) syntax elements describe how the prediction is performed in order to reconstruct the samples belonging to each PU. Coding efficiency improvements have been made in HEVC for both modeling and coding of motion parameters and intra prediction modes. While H.264/AVC uses a *single* motion vector predictor (unless direct mode is used) and a *single* most probable mode (MPM), HEVC uses *multiple* candidate predictors or MPMs together with an index or flag for signaling the selected predictor or MPM, respectively. In addition, HEVC provides a mechanism for exploiting spatial and temporal dependencies with regard to motion modeling by *merging* neighboring blocks with identical motion parameters. This has been found to be particularly useful in combination with quadtree-based block partitioning, since a pure hierarchical subdivision approach may lead to partitionings with suboptimal rate-distortion behavior [32, 49, 102]. Also, due to the significant increased number of angular intra prediction modes relative to H.264/AVC, three MPMs for each PU are considered in HEVC.

This section will discuss how the various PU syntax elements are processed in terms of binarization, context modeling, and context assignment. Also, aspects related to parsing dependencies and throughput for the various prediction parameters are considered.

8.5.1 Motion Data Coding

In HEVC, motion data can be either signaled using merge mode or directly using motion vector differences, reference indices, and inter-prediction direction.

8.5.1.1 Signaling of Merge Mode

In HEVC, merge mode enables motion data (i.e., prediction direction, reference index and motion vectors) to be inherited from a spatial or temporal (co-located) neighbor. A list of merge candidates are generated from these neighbors. `merge_flag` is signaled to indicate whether merge is used in a given PU. If merge is used, then `merge_idx` is signaled to indicate from which candidate the motion data should be inherited. `merge_idx` is coded with truncated unary, which means that the bins are parsed until a zero bin value is reached or when the number of bins is equal to the cMax, the max allowed number of bins.

Determining how to set cMax involved evaluating the throughput and coding efficiency trade-offs in a core experiment [7]. For optimal coding efficiency, cMax should be set to equal the merge candidate list size of the PU. Furthermore, `merge_flag` should not be signaled if the list is empty. However, this makes parsing depend on list construction, which is needed to determine the list size.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00245

Constructing the list requires a large amount of computation since it involves reading from multiple locations (i.e., fetching the co-located neighbor and spatial neighbors) and performing several comparisons to prune the list; thus, dependency on list construction would significantly degrade parsing throughput [33, 108].

To decouple the list generation process from the parsing process such that they can operate in parallel in HEVC, cMax is signaled in the slice header using `five_minus_max_num_merge_cand` and does not depend on list size. To compensate for the coding loss due to the fixed cMax, combined bi-predictive and zero motion vector candidates are added when the list size is less than the maximum number of allowed candidates as defined by cMax [79]. This also ensures that the list is never empty and that `merge_flag` is always signaled [107]. For more details on candidate list construction please refer to Chap. 5.

8.5.1.2 Signaling of Motion Vector Differences, Reference Indices, and Inter-Prediction Direction

If merge mode is not used, then the motion vector is predicted from its neighboring blocks and the difference between motion vector (mv) and motion vector prediction (mvp), referred to as *motion vector difference* (mvd), is signaled:

$$\text{mvd}(x, y) = \text{mv}(x, y) - \text{mvp}(x, y)$$

In H.264/AVC, a single predictor is calculated for mvp from the median of the left, top and top-right spatial 4×4 neighbors.

In HEVC, advanced motion vector prediction (AMVP) is used, where several candidates for mvp are determined from spatial and temporal neighbors [38]. A list of mvp candidates is generated from these neighbors, and the list is pruned to remove redundant candidates such that there is a maximum of two candidates. A syntax element called `mvp_l0_flag` (or `mvp_l1_flag` depending on the reference list) is used to indicate which candidate is used from the list as the mvp. To ensure that parsing is independent of list construction, `mvp_l0_flag` is signaled even if there is only one candidate in the list. The list is never empty as the zero motion vector is used as the default candidate.

In HEVC, improvements were also made on the coding process of mvd itself. In H.264/AVC, the first nine bins of mvd are regular coded truncated unary bins, followed by bypass coded 3rd order Exp-Golomb bins. In HEVC, the number of regular coded bins for mvd is significantly reduced [58]. Only the first two bins are regular coded (`abs_mvd_greater0_flag`, `abs_mvd_greater1_flag`), followed by bypass coded first-order Exp-Golomb (EG1) bins (`abs_mvd_minus2`).

In H.264/AVC, context selection for the first bin in mvd depends on whether the sum of the motion vectors of the top and left 4×4 neighbors are greater than 32 (or less than 3). This requires 5-bit storage per neighboring motion vector, which accounts 24,576 of the 30,720-bit CABAC line buffer needed to support a 4k ×2k sequence. The need to reduce the line buffer size in HEVC by modifying

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00246

the context selection logic was highlighted in [90]. Accordingly, all dependencies on the neighbors were removed and the context is selected based on the binIdx (i.e., whether it is the first or second bin) [82, 91].

To maximize the impact of fast bypass coding, the bypass coded bins for both the horizontal (x) and vertical (y) components of mvd are grouped together in HEVC [67]. For instance, for a motion vector difference of $\text{mvd}(x, y) = (2, 2)$, the coding order is 1111$\underline{0000}$, where underlined values are bypass coded. Without bypass grouping, the coding order is 11$\underline{00}$11$\underline{00}$. If four bypass bins can be processed in a single cycle, enabling bypass grouping reduces the number of cycles required to process the motion vector by one.

In HEVC, reference indices `ref_idx_l0` and `ref_idx_l1` are coded with truncated unary regular coded bins, which is the same as for H.264/AVC; the maximum length of the truncated unary binarization, cMax, is dictated by the reference picture list size. However, in HEVC only the first two bins are regular coded [71], whereas all bins are regular coded in H.264/AVC. In both HEVC and H.264/AVC, the regular coded bins of the reference indices for different reference picture lists share the same set of contexts. The inter-prediction direction (list 0, list 1 or bi-directional) is signaled using `inter_pred_idc` with custom binarization.

8.5.2 *Intra Prediction Mode Coding*

Similar to motion data coding, a most probable mode (MPM) is calculated for intra mode coding. In H.264/AVC, the minimum mode of the top and left neighbors is used as MPM. `prev_intra4x4_pred_mode_flag` (or `prev_intra8x8_pred_mode_flag`) is signaled to indicate whether the most probable mode is used. If the MPM is not used, the remainder mode `rem_intra4x4_pred_mode_flag` (or `rem_intra8x8_pred_mode_flag`) is signaled.

In HEVC, additional MPMs are used to improve coding efficiency. A candidate list of most probable modes with a fixed length of three is constructed based on the left and top neighbors. The additional candidate modes (DC, planar, vertical) can be added if the left and top neighbors are the same or unavailable. Note that the top neighbors outside the current CTU are considered unavailable in order to avoid the need for a line buffer.[14] The prediction flag `prev_intra_pred_mode_flag` is signaled to indicate whether one of the most probable modes is used. If an MPM is used, a most probable mode index (`mpm_idx`) is signaled to indicate which candidate to use. It should be noted that in HEVC, the order in which the coefficients of the residual are parsed (e.g., diagonal, vertical or horizontal) depends on the reconstructed intra mode (i.e., the parsing of the TU data that follows depends on list construction and intra mode reconstruction). Thus, the candidate list size was limited to three for reduced computation to ensure that it would not affect entropy decoding throughput [22, 83].

[14] For more details on MPM list construction please refer to Chap. 4.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00247

Table 8.5 Differences between prediction unit coding in HEVC and H.264/AVC

	HEVC			H.264/AVC	
Properties	Intra Mode	AMVP	Merge	Intra Mode	MVP
Max number of candidates in list	3	2	5	1	1
Spatial neighbor	used	used	used	used	used
Temporal co-located neighbor	not used	used	used	not used	not used
Number of contexts	2	10	2	6	20
Max regular coded bins per PU	2	16	2	7	98

The number of regular coded bins was reduced for intra mode coding in HEVC relative to the corresponding part in H.264/AVC, where both the flag and the 3 fixed-length bins of the remainder mode are regular coded using two separate context models. In HEVC, the flag is regular coded as well, but the remainder mode is a fixed-length 5-bin value that is entirely bypass coded. The most probable mode index (`mpm_idx`) is also entirely bypass coded. The number of contexts used to code `intra_chroma_pred_mode` is reduced from 4 to 1 for HEVC relative to H.264/AVC. To maximize the impact of fast bypass coding, the bypass coded bins for luma intra prediction mode coding within a CU are grouped together in HEVC [19]. This is beneficial when the partition mode is PART_NxN, and there are four sets of prediction modes.

8.5.3 Comparison of HEVC and H.264/AVC

The differences between H.264/AVC and HEVC in signaling of syntax elements at the PU layer are summarized in Table 8.5. HEVC uses both spatial and temporal neighbors as predictors, while H.264/AVC only uses spatial neighbors (unless direct mode is enabled). In terms of the impact of the throughput improvement techniques, HEVC has around 6× fewer maximum regular coded bins per inter-predicted PU than H.264/AVC. HEVC also requires around 2× fewer contexts for PU syntax elements than H.264/AVC.

8.6 Transform Unit Syntax Elements

In video coding, both intra and inter prediction are used to reduce the amount of data that needs to be transmitted. In addition, rather than sending the original samples of the prediction signal, an appropriately quantized approximation of the prediction error is transmitted. To this end, the prediction error is blockwise transformed

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00248

Table 8.6 Distribution of bins in CABAC for HEVC and H.264/AVC under common test conditions [6, 101] and for the worst case

	HEVC					H.264/AVC		
Common conditions	AI MAIN	LP MAIN	LB MAIN	RA MAIN	worst case	HierB	HierP	worst case
CTU/CU bins	5.4%	15.8%	16.7%	11.7%	1.4%	27.0%	34.0%	0.5%
PU bins	9.2%	20.6%	19.5%	18.8%	5.0%	23.4%	26.3%	15.8%
TU bins	85.4%	63.7%	63.8%	69.4%	94.0%	49.7%	39.7%	83.7%

Generated bins are discriminated along the HEVC categories CTU/CU, PU, and TU as well as their corresponding counterparts in H.264/AVC

from spatial to frequency domain, thereby decorrelating the residual samples and performing an energy compaction in the sense that, after quantization, the signal can be represented in terms of a few non-vanishing coefficients. The method of signaling the quantized values and frequency positions of these coefficients is referred to as *transform coefficient coding*.

Syntax elements related to transform coefficient coding account for a significant portion of the bin workload as shown in Table 8.6. At the same time, those syntax elements also account for a significant portion of the total number of bits for a compressed video, and as a result the compression of quantized transform coefficients significantly impacts the overall coding efficiency. Thus, transform coefficient coding with CABAC must be carefully designed in order to balance coding efficiency and throughput demands. Accordingly, as part of the HEVC standardization process, a core experiment on coefficient scanning and coding was established to investigate tools related to transform coefficient coding [97].

This section describes how transform coefficient coding evolved from H.264/AVC to the first test model of HEVC (HM1.0) to the Final Draft International Standard (FDIS) of HEVC (HM10.0), and discusses the reasons behind design choices that were made. Many of the throughput improvement techniques were applied, and new tools for improved coding efficiency were simplified. As a reference for the beginning and end points of the development, Figs. 8.8 and 8.9 show examples of transform coefficient coding for 4×4 blocks in H.264/AVC and HEVC, respectively.

8.6.1 Transform Block Structure

As already discussed in Sect. 8.3.1, transform coding in HEVC involves a tree-structured variable block-size approach with supported transform block sizes of 4×4, 8×8, 16×16, and 32×32. This means that the actual transform block sizes, used to code the prediction error of a given CU, can be selected based on the characteristics of the residual signal by using a quadtree-based partitioning, also known as residual quadtree (RQT), as illustrated in Fig. 8.10. While this larger

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00249

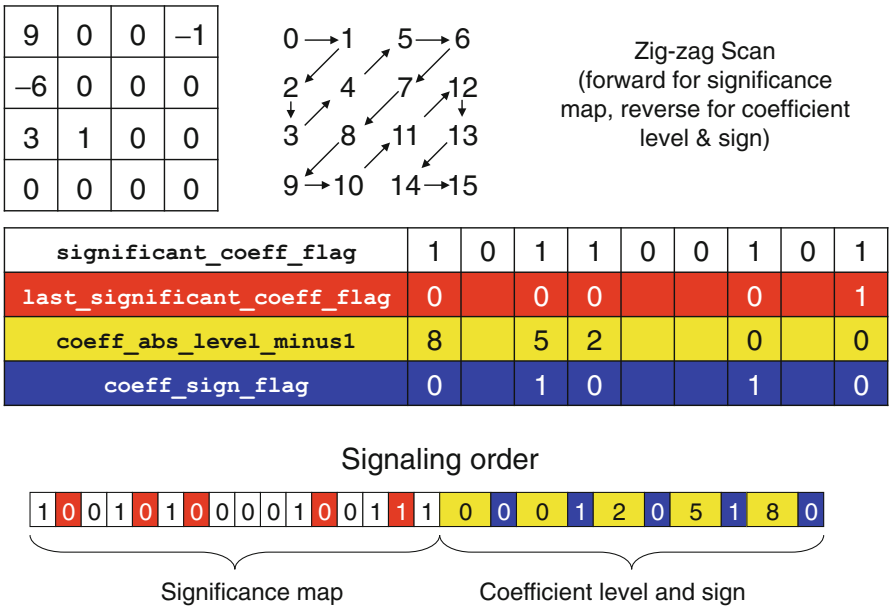


Fig. 8.8 Example of CABAC-based transform coefficient coding for a 4 × 4 transform block in H.264/AVC. Note, however, that the corresponding bins for signaling of the absolute level (in *yellow*) are not explicitly shown

variety of transform block partitioning relative to H.264/AVC provides significant coding gains, it also has implications in terms of implementation costs, both in terms of memory bandwidth and computational complexity. To address this issue, HEVC allows to restrict the RQT-based transform block partitioning by four parameters, signaled by corresponding syntax elements in the SPS: the maximum and minimum allowed transform block size (in terms of block width) $n_{\max}$ and $n_{\min}$, respectively, and the maximum depth of the RQT $d_{\max}$, with the latter given both individually for intra-picture and inter-picture prediction. Note, however, that there is a rather involved interdependency between these parameters (and other syntax elements), such that, for instance, implicit subdivisions or implicit leaf nodes of the RQT may occur. For more details, please refer to Chap. 3.

The signaling of the transform block structure for each CU is similar to that of the coding block structure at the CTU level. For each node of the RQT, a flag called `split_transform_flag` is signaled to indicate whether a given transform block should be further subdivided into four smaller TBs. Context modeling for the coding of this flag involves three different contexts with its related context increment equal to $5 - \log_2(\text{TrafoSize})$, where TrafoSize denotes the block width of the corresponding luma transform block at the given RQT depth. Note that for the choice of a luma CTB size of 64, $n_{\max} = 32$, $n_{\min} = 4$, and $d_{\max} = 4$, an implicit leaf node is implied for the case of TrafoSize $= 4$, whereas an implicit

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00250

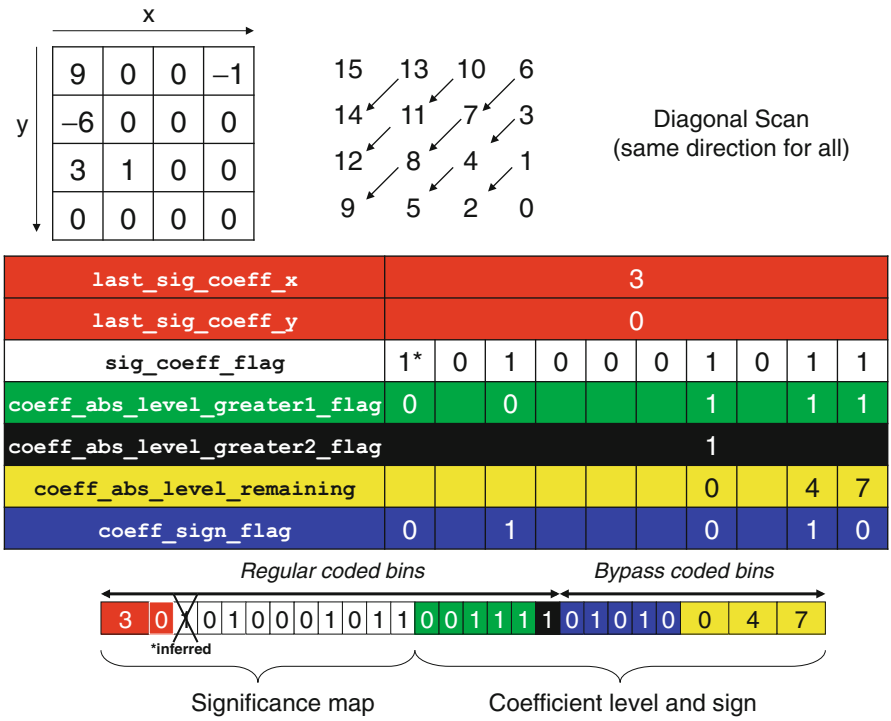


Fig. 8.9 Example of transform coefficient coding for a 4 × 4 transform block in HEVC. Note, however, that the corresponding bins for signaling of the "last" information (in *red*) and absolute level remaining (in *yellow*) are not explicitly shown

subdivision is given for a luma CB size of 64 at RQT depth equal to 0. Table 8.7 and Fig. 8.11 illustrate an example of this configuration. Therefore, even if up to five different RQT levels are permitted, only up to three different context models are required for coding of `split_transform_flag`. Note that the signaling of `split_transform_flag` at the RQT root is omitted if the quantized residual of the corresponding CU contains no non-zero transform coefficient at all, i.e., if the corresponding coded block flag at the RQT root (see Sect. 8.6.3) is equal to 0.

8.6.2 Transform Skip

For regions or blocks with many sharp edges (e.g., as typically given in screen content coding), coding gains can be achieved by skipping the transform [42, 61]. When the transform is skipped for a given block, the prediction error in the spatial domain is quantized and coded in the same manner as for transform coefficient coding (i.e., the quantized block samples of the spatial error are coded as if they were quantized transform coefficients). The so-called *transform skip* mode is only

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00251

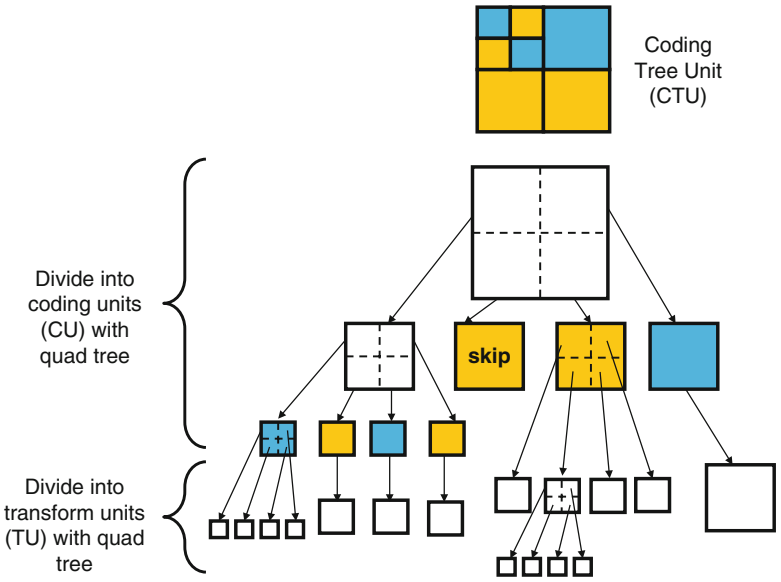


Fig. 8.10 Illustration of residual quadtrees (one for each CU) used to signal transform units for residual coding of CUs. Note that the same relationships and comments as given in Fig. 8.6 apply here as well

allowed for 4 × 4 TUs and only if the corresponding functionality is enabled by the `transform_skip_enabled_flag` in the PPS. Signaling of this mode is performed by using the `transform_skip_flag`, which is coded using a single fixed context model.

8.6.3 Coded Block Flags

At the top level of the hierarchy of significance flags, as already explained in Sect. 8.3.1, coded block flags (CBFs) are signaled for the RQT root, i.e., at the CU level in the form of the `rqt_root_cbf` and for subsequent luma and chroma TBs in the form of `cbf_luma` and `cbf_cb`, `cbf_cr`, respectively. `rqt_root_cbf` is only coded and transmitted for inter-predicted CUs that are not coded in merge mode using a single PU (PART_2Nx2N)[15]; for that a single context model is used. While signaling of `cbf_luma` is only performed at the leaf nodes of the RQT, provided that a non-zero `rqt_root_cbf` was signaled before, the chroma CBFs `cbf_cb` and `cbf_cr` are transmitted at each internal node as long as a corresponding non-zero chroma CBF at its parent node occurred. For coding of both

[15]Intra-predicted CUs typically have nonzero residual, so `rqt_root_cbf` is not used.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00252

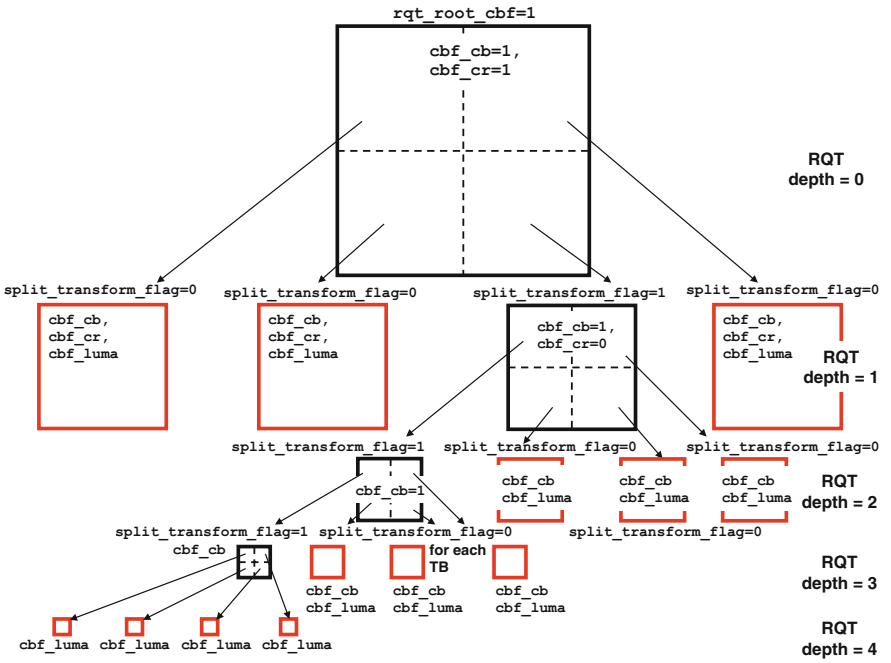


Fig. 8.11 Illustration of signaling of `split_transform_flag`, `cbf_luma`, `cbf_cb`, and `cbf_cr` for an RQT with depth 4. Note that at RQT depth = 0, no `split_transform_flag` is signaled since an implicit transform split occurs for CU of 64 as $n_{\max} = 32$. `cbf_luma` is only signaled for leaf transform blocks (highlighted in *red*). `cbf_cb` and `cbf_cr` are signaled for the root node and all nodes where the corresponding CBF at the parent node is non-zero, except for the nodes related to TrafoSize = 4

`cbf_cb` and `cbf_cr`, four contexts are used such that the corresponding context increment depends on the RQT depth (with admissible values between 0 and 3, since for the case of TrafoSize = 4 no chroma CBFs are transmitted), whereas for `cbf_luma` only two contexts are provided with its discriminating context increment depending on the condition RQT depth = 0. For more background on the use of RQT and related syntax elements, please refer to Chap. 3.

8.6.4 Significance Map

In H.264/AVC, the significance map for each transform block is signaled by transmitting a `significant_coeff_flag` (SIG) for each position to indicate whether the coefficient is non-zero. The positions are processed in an order based on a zig-zag scan. After each non-zero SIG, an additional flag called `last_significant_coeff_flag` (LAST) is immediately sent to indicate whether

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00253

Table 8.7 Derivation of context increment (ctxInc) for `split_transform_flag`, `cbf_luma`, `cbf_cb`, and `cbf_cr` for the example in Fig. 8.11

RQT depth	Transform Size	`split_transform_flag` (ctxInc)	`cbf_luma` (ctxInc)	`cbf_cb`, `cbf_cr` (ctxInc)
0	n/a	n/a	1	0
1	32 × 32	0	0	1
2	16 × 16	1	0	2
3	8 × 8	2	0	3
4	4 × 4	n/a	0	n/a

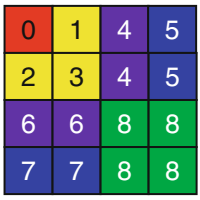


Fig. 8.12 Context index assignment for `sig_coeff_flag` in 4 × 4 TBs

it is the last non-zero SIG; this prevents unnecessary SIG from being signaled. Different contexts are used depending on the position within the 4 × 4 and 8 × 8 transform blocks, and whether the bin represents an SIG or LAST. Since SIG and LAST are interleaved, the context selection of the current bin depends on the immediate preceding bin. The dependency of LAST on SIG results in a strong bin to bin dependency for context selection of significance map entries in H.264/AVC as illustrated in Fig. 8.3.

8.6.4.1 `sig_coeff_flag` (SIG)

While in HEVC position based context assignment for coding of `sig_coeff_flag` (SIG) is used for 4 × 4 TBs as shown in Fig. 8.12, new forms of context assignment for larger transforms were needed. In HM1.0, additional dependencies were introduced in the context selection of SIG for 16 × 16 and 32 × 32 TBs to improve coding efficiency. Specifically, the context selection of SIG was calculated based on a local template using 10 (already decoded) SIG neighbors as shown in Fig. 8.13a [57, 102]. By using this template-based context selection bit rate savings of 1.4–2.8 % were reported [57].

To reduce context selection dependencies and storage costs, Sze and Budagavi [85] proposed using fewer neighbors and showed that this could be done without severely sacrificing coding efficiency. For instance, using only a maximum of 8 neighbors (removing neighbors A and D as shown in Fig. 8.13b) had negligible impact on coding efficiency, while using only six neighbors (removing neighbors A, B, D, E and H as shown in Fig. 8.13c) results in a coding efficiency loss of only 0.2 %. This was further extended in [18] for HM2.0, where only a maximum of five neighbors was used by removing dependencies on positions G and K, as

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00254

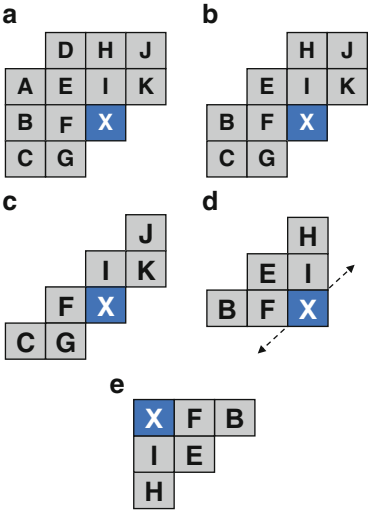


Fig. 8.13 Local templates for SIG context selection. X (in *blue*) represents the current position of the bin being processed. (**a**) Ten neighbors (HM1.0), (**b**) eight neighbors, (**c**) six neighbors, (**d**) 5 neighbors (HM3.0), and (**e**) inverted for reverse scan (HM4.0)

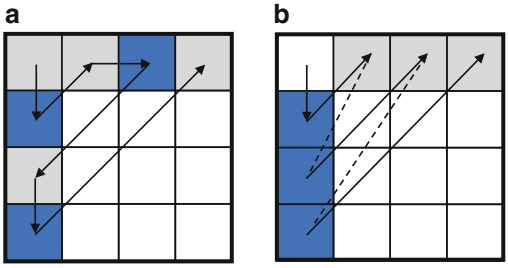


Fig. 8.14 Scans used to process SIG. Diagonal scan avoids dependency on the most recently processed bin. Context selection for *blue* positions is affected by values of the neighboring *grey* positions. (**a**) Zig-zag scan and (**b**) diagonal scan

shown in Fig. 8.13d. In HM2.0, the significance map was scanned in zig-zag order, so removing the diagonal neighbors G and K is important since those neighbors pertain to the most recently decoded SIG.

Despite reducing the number of SIG neighbors in HM2.0, dependency on the most recently processed SIG neighbors still existed for the positions at the edge of the transform block as shown in Fig. 8.14a. The horizontal or vertical shift that is required to go from one diagonal to the next in the zig-zag scan causes the previously decoded bin to be one of the neighbors (F or I) that is needed for context selection. In order to address this issue, in HM4.0, a diagonal scan was introduced to replace the zig-zag scan [86] as shown in Fig. 8.14b. Changing from zig-zag to diagonal scan had negligible impact on coding efficiency, but removed the dependency on recently processed SIG for all positions in the TB. In HM4.0, the scan was also reversed (from high frequency to low frequency) [74]. Accordingly, the neighbor dependencies were inverted from top-left to bottom-right, as shown in Fig. 8.13e.

Dependencies in context selection of SIG for 16 × 16 and 32 × 32 TBs were further reduced in HM7.0, where 16 × 16 and 32 × 32 TBs are divided into 4 × 4 subblocks. This will be described in more detail in Sect. 8.6.4.3 on

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00255

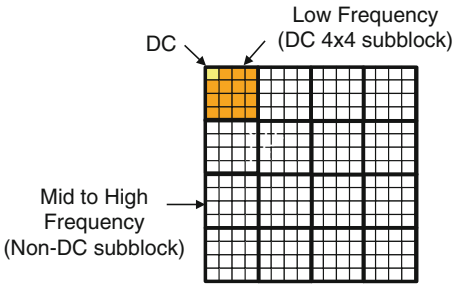


Fig. 8.15 Regions in 8 × 8, 16 × 16 and 32 × 32 TBs map to different context sets for SIG

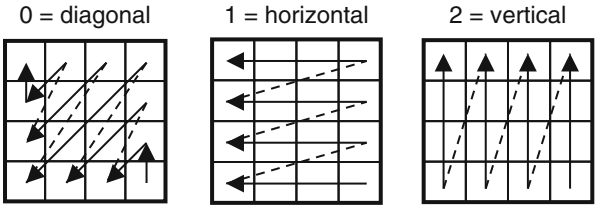


Fig. 8.16 Diagonal, vertical, and horizontal scans for 4 × 4 TBs

Table 8.8 Mode dependent coefficient scanning: Mapping of intra prediction mode to scans (0 = Diagonal, 1 = Horizontal, 2 = Vertical) for different TB sizes and components

Intra Prediction Mode	0 (Planar)	1 (DC)	2	3	4	5	6 to 14	15 to 21	22 to 30	31 to 34
8 × 8 (luma)	0					0	2	0	1	0
4 × 4 (luma or chroma) TB	0					0	2	0	1	0
Otherwise	0									

`coded_sub_block_flag` (CSBF). In HM8.0, 8 × 8 TBs were also divided into 4 × 4 subblocks such that all TB sizes above 4 × 4 are based on a 4 × 4 subblock processing for a harmonized design [77].

The 8 × 8, 16 × 16 and 32 × 32 TBs are divided into three regions based on frequency, as shown in Fig. 8.15. The DC, low-frequency and mid/high-frequency regions all use different sets of contexts. To reduce memory size, the contexts for coding the SIG of 16 × 16 and 32 × 32 TBs are shared [81, 99].

For improved coding efficiency for intra predicted CUs, so-called mode dependent coefficient scanning (MDCS) is used to select between vertical, horizontal, and diagonal scans based on the chosen intra prediction mode [106], as illustrated in Fig. 8.16. Table 8.8 shows how the scans are assigned based on intra prediction mode, TB size, and component. As mentioned in Sect. 8.5.2, this requires the intra mode to be decoded before decoding the corresponding transform coefficients. MDCS is only used for 4 × 4 and 8 × 8 TBs and provides coding gains of up to 1.2 %. Note that for TBs larger than 8 × 8, and for TBs of inter predicted CUs, only the diagonal scan is used.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00256

Table 8.9 Binarization of coordinate values of the last position

Coordinate Value	TB $= 4 \times 4$	TB $= 8 \times 8$	TB $= 16 \times 16$	TB $= 32 \times 32$
0	0	0	0	0
1	10	10	10	10
2	110	110	110	110
3	111	1110	1110	1110
4	n/a	11110 0	11110 0	11110 0
5	n/a	11110 1	11110 1	11110 1
6	n/a	11111 0	111110 0	111110 0
7	n/a	11111 1	111110 1	111110 1
8	n/a	n/a	1111110 00	1111110 00
9	n/a	n/a	1111110 01	1111110 01
10	n/a	n/a	1111110 10	1111110 10
11	n/a	n/a	1111110 11	1111110 11
12-15	n/a	n/a	1111111 xx	11111110 xx
16-23	n/a	n/a	n/a	111111110 xxx
24-31	n/a	n/a	n/a	111111111 xxx

Bins belonging to the bypass coded suffixes are underlined

8.6.4.2 Last Position Coding

As mentioned earlier, there are strong data dependencies between `significant_coeff_flag` (SIG) and `last_significant_coeff_flag` (LAST) in H.264/AVC due to the fact that they are interleaved. Budagavi and Demircin [10] proposed grouping several SIG together by transmitting a LAST only once per N number of SIG. If all of the N SIG are zero, LAST is not transmitted. Sole et al. [73] avoids interleaving of SIG and LAST altogether. Specifically, the horizontal (x) and vertical (y) position of the last non-zero SIG in a TB is sent in advance rather than LAST by using the syntax elements `last_sig_coeff_x` and `last_sig_coeff_y`, respectively. For instance, in the example shown in Fig. 8.9, `last_sig_coeff_x` equal to 3 and `last_sig_coeff_y` equal to 0 are sent before processing the TB rather than signaling LAST for each SIG with value of 1. Signaling the (x, y) position of the last non-zero SIG for each TB was adopted into HM3.0. Note that the SIG in the last scan position is inferred to be 1.

The last position, given by its coordinates in both x and y direction, is composed of a prefix and suffix as shown in Table 8.9. The prefixes `last_sig_coeff_x_prefix` and `last_sig_coeff_y_prefix` are both regular coded using TrU binarization with cMax $= 2 * (\log_2 \text{TrafoSize}) - 1$ [70]. A suffix is present when the corresponding prefix is composed of more than four bins. In that case, the suffixes `last_sig_coeff_x_suffix` and `last_sig_coeff_y_suffix` are bypass coded using FL binarization. Some of the contexts are shared across the chroma TB sizes to reduce context memory, as shown in Table 8.10. To maximize the impact of fast bypass coding, the bypass coded bins (i.e., the suffix bins) for both the x and y coordinate of the last position are grouped together for each TB in HEVC.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00257

Table 8.10 Context selection for regular coded prefix bins of the coordinates of the last position `last_sig_coeff_x_prefix` and `last_sig_coeff_y_prefix`

Bin Index	0	1	2	3	4	5	6	7	8
4 × 4 luma TB	0	1	2						
8 × 8 luma TB	3	3	4	4	5				
16 × 16 luma TB	6	6	7	7	8	8	9		
32 × 32 luma TB	10	10	11	11	12	12	13	13	14
4 × 4 chroma TB	15	16	17						
8 × 8 chroma TB	15	15	16	16	17				
16 × 16 chroma TB	15	15	15	15	16	16	16		

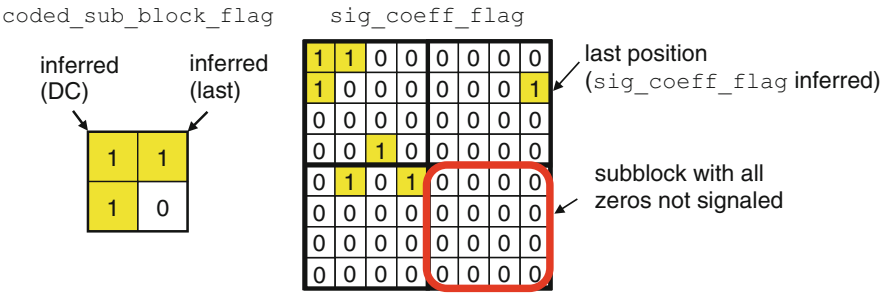


Fig. 8.17 Example of the hierarchical signaling of an 8 × 8 significance map

8.6.4.3 `coded_sub_block_flag` (CSBF)

As already explained in Sect. 8.3.1, the number of bins to be transmitted for signaling the significance map is considerably reduced by using a hierarchical signaling scheme of significance flags. Part of this hierarchy is the `coded_sub_block_flag` (CSBF) that indicates for each 4 × 4 subblock of a TB whether there are non-zero coefficients in the subblock [56, 60]. If CSBF is equal to 1, the subblock contains at least one non-zero transform coefficient level and, consequently, SIGs within the subblock are signaled. No SIGs are signaled for a 4 × 4 subblock that contains all vanishing transform coefficients, since this information is signaled by a CSBF equal to 0. For large TB sizes, a reduction in SIG bins of up to a 30 % can be achieved by the use of CSBFs, which corresponds to an overall bin reduction of 3–4 % under common test conditions. To avoid signaling of redundant information, the CSBF for the subblocks containing the DC and the last position are inferred to be equal to 1. Figure 8.17 shows an example of the hierarchical signaling of an 8 × 8 significance map.

In HM7.0, the CSBF was additionally used to further reduce dependencies in the context selection of SIG for 16 × 16 and 32 × 32 TBs. Specifically, the neighboring subblocks and their corresponding CSBFs (Fig. 8.18) are used for context selection rather than the individual SIG neighbors, as shown in Fig. 8.13e [41]. This context selection scheme was extended to 8 × 8 TBs in HM8.0 [77]. According to this

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00258

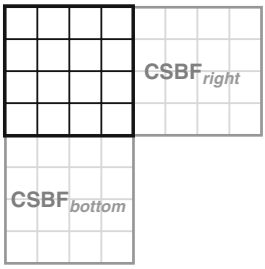


Fig. 8.18 Neighboring CSBFs (right, bottom) used for SIG context selection

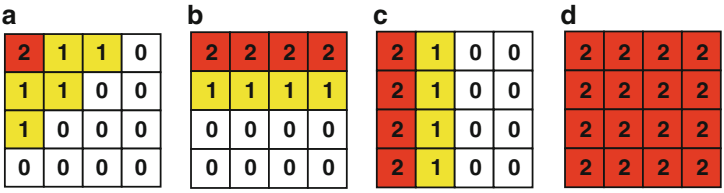


Fig. 8.19 4 × 4 position based mapping for SIG context selection based on CSBF of neighboring subblocks. (**a**) Pattern 1, (**b**) pattern 2, (**c**) pattern 3 and (**d**) pattern 4

scheme, the CSBF of the neighboring right and bottom subblocks (CSBF$_{right}$, CSBF$_{bottom}$) are used to select one of four patterns shown in Fig. 8.19: (0,0) maps to pattern 1, (1,0) to pattern 2, (0,1) to pattern 3 and (1,1) to pattern 4. The pattern maps each position within the 4 × 4 subblock to one of three contexts. As a result, there are no intrinsic dependencies for context selection of SIG within each 4 × 4 subblock.

Reverse diagonal scanning order is used within the subblocks and for the processing order of the subblocks themselves, as shown in Fig. 8.20 [76]. Both significance map and coefficient levels are processed in this order. As an exception to this rule, for 4 × 4 and 8 × 8 TBs to which MDCS is applied, reverse vertical and horizontal scanning orders are used within the subblocks as well as for the processing order of the subblocks themselves. Furthermore, as shown in Table 8.11, different sets of contexts for coding of SIG are used for diagonal and non-diagonal (vertical and horizontal) scans in both 4 × 4 luma and chroma TBs, and 8 × 8 luma TBs [77].

8.6.4.4 Summary of Significance Map Coding in HEVC

Figure 8.21 summarize the steps required to code the significance map. This process is repeated for every non-zero TB in HEVC. Table 8.11 summarizes the multiple steps of classification used to assign the 42 contexts of `sig_coeff_flag`. Contexts 0 to 26 are used for luma coded TBs, while 27 to 41 used for chroma TBs. The contexts are further mapped based on the TB size, the scan direction, whether the subblock is DC or non-DC, CSBF of neighboring subblocks, and position within

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00259

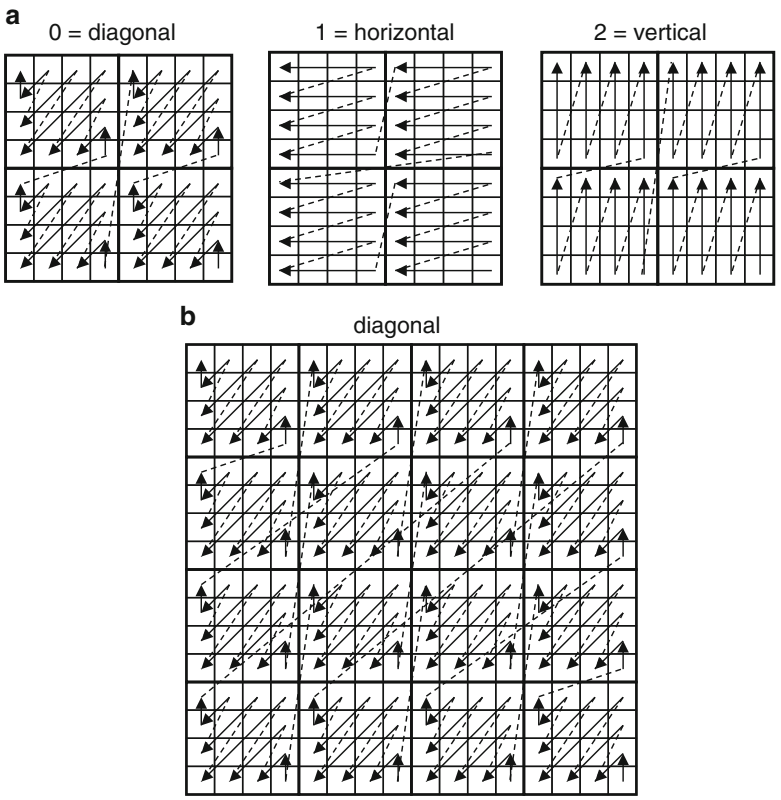


Fig. 8.20 Subblock scans. Scan for 4 × 4 TB shown in Fig. 8.16. (**a**) Subblock scan for 8×8 TB. (**b**) Subblock scan for 16 × 16 TB. Scan for 32 × 32 TB is also all diagonal

Table 8.11 Context selection of `sig_coeff_flag` based on component, TB size, scan order (Table 8.8), position of subblock within the TB (Fig. 8.15), and position based context index within 4 × 4 TB or subblock (SubIdx) (Fig. 8.12 or 8.19, resp.)

ctxIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
Component	luma																										
TB size	All	4×4								8×8												16×16 & 32×32					
Scan	All	All								Diagonal						Vertical/Horizontal						Diagonal					
Subblock position	DC	Only one subblock								DC subblock			Non-DC subblock			DC subblock			Non-DC subblock			DC subblock			Non-DC subblock		
SubIdx	DC	1	2	3	4	5	6	7	8	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

ctxIdx	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**
Component	chroma														
TB size	All	4×4								8×8			16×16		
Scan	All	All scans								Diagonal					
Subblock position	DC	Only one subblock								All subblocks			All subblocks		
SubIdx	DC	1	2	3	4	5	6	7	8	0	1	2	0	1	2

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00260

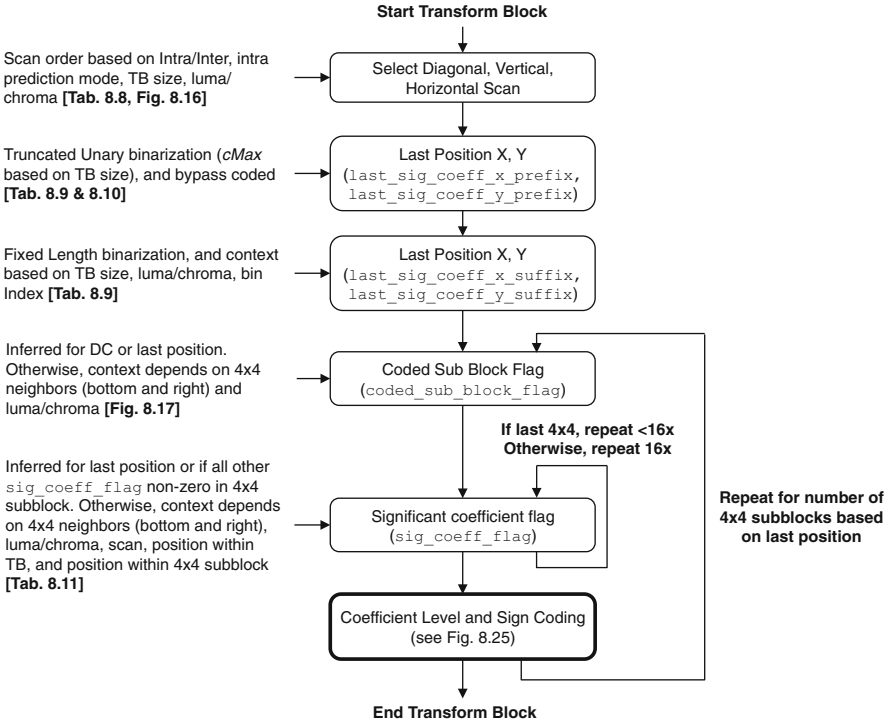


Fig. 8.21 Flow chart for coding the syntax elements of a TB in HEVC

the subblock. Note that context 0 is used to code the `sig_coeff_flag` of the DC position of all luma TBs, and context 27 is used for the DC position of all chroma TBs.

8.6.5 Absolute Coefficient Level and Coefficient Sign

In HEVC, parsing of transform coefficient level information is performed subblock-by-subblock using up to five scan passes for each subblock. The first scan pass is devoted to the SIG flags, as already explained in Sects. 8.6.4.1 and 8.6.4.3. In the second and third pass, the two additional flags `coeff_abs_level_greater1_flag` (ALG1) and `coeff_abs_level_greater2_flag` (ALG2) are conditionally parsed, indicating for each relevant scan position if the corresponding absolute value of the coefficient level, i.e., the absolute level (AL) is greater than 1 and 2, respectively. However, only up to 8 ALG1 flags and one ALG2 flag are transmitted for each subblock, as will be explained in more detail below. In the third scan pass, the sign of each significant level is signaled with the possible exception

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00261

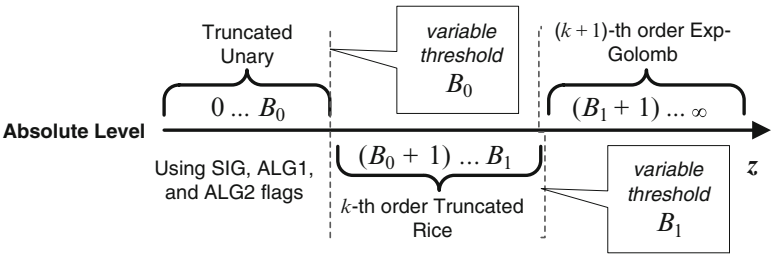


Fig. 8.22 Illustration of the adaptive binarization scheme for absolute levels in HEVC consisting of a concatenation of the three elementary binarizations TrU, TRk, and EGk, the latter two with varying order k and $k+1$, respectively ($0 \leq k \leq 4$). The two variable thresholds B_0 and B_1 specify the (variable) transition points between them

of the last non-zero coefficient in the subblock in reverse scanning order, as will be discussed in more detail in Sect. 8.6.5.2. Finally, in the last and fifth scan pass, the remaining information of absolute levels in the subblock (if present) is transmitted by using the syntax element `coeff_abs_level_remaining` (ALRem), as will be further detailed in Sect. 8.6.5.1 below.

8.6.5.1 Coding of Absolute Level

Coding of absolute levels requires the choice of suitable binarization schemes and, for selected bin indices, the choice of suitable context models. According to the design considerations, as discussed in Sect. 8.3, both aspects of coding efficiency and throughput have been properly addressed by the revised CABAC design of HEVC. This is especially true for the coding of absolute levels which typically contribute the dominant portion to the total number of generated bins. In the following, we will first elaborate on how the specific binarization scheme for absolute levels in HEVC has been designed. Then, in the second part of this subsection, we will present the context selection rules applied to the (few remaining) regular coded bins of absolute levels, unless not already done so in Sects. 8.6.4.1 and 8.6.4.3.

Conceptually, the binarization of an absolute level, denoted as z in the following, relies on a concatenated application of three binarization processes [21, 54, 59]: truncated unary (TrU), k-th order truncated Rice (TRk), and $(k+1)$-th order Exp-Golomb (EGk). Figure 8.22 illustrates this binarization scheme for arbitrary z along the (discrete) number line. There are two thresholding parameters B_0, B_1 with $B_0 < B_1$ which separate the three regions from one another for application of each of the three binarization processes and which also determine the truncation parameters $\mathrm{cMax(TrU)} = B_0 + 1$ and $\mathrm{cMax(TRk)} = B_1 - B_0$. The selection of the two parameters B_0, B_1 together with the choice of the parameter k is performed in a backward-adaptive manner for each subblock in such a way that the resulting bin strings are already close to a minimum-redundancy prefix code for the collection of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00262

Table 8.12 Binarization of the absolute level z for the choice of parameters $B_0 = 2$, $B_1 = 6$, and $k = 0$, corresponding to a concatenation of TrU with cMax = 3, zero-order Truncated Rice (TRk) with cMax = 4, and first-order Exp-Golomb (EGk)

		TrU			TRk				EGk				
		cMax=3			$k = 0$; cMax=4				$k + 1 = 1$				
z		SIG	ALG1	ALG2	0	1	2	3	0	1	2	3	...
	0	0											
	1	1	0										
B_0 =	2	1	1	0									
	3	1	1	1	0								
	4	1	1	1	1	0							
	5	1	1	1	1	1	0						
B_1 =	6	1	1	1	1	1	1	0					
	7	1	1	1	1	1	1	1	0	0			
	8	1	1	1	1	1	1	1	0	1			
	9	1	1	1	1	1	1	1	1	0	0	0	
	10	1	1	1	1	1	1	1	1	0	0	1	
	11	1	1	1	1	1	1	1	1	0	1	0	
	...	...	...	...	...	...	...	...	...	...	...	...	...

all absolute levels z in each subblock. As a consequence, the majority of resulting bins can be simply bypass coded without compromising coding efficiency.

For each subblock, the initialization and adaptation processes for the parameters B_0, B_1, and k is performed, as follows. Before starting the processing of an subblock, the parameter k is set equal to 0, whereas B_0 is set equal to 2. The second thresholding parameter B_1 depends on k and B_0 by the fixed relation $B_1 = 4*2^k + B_0$, which means that B_1 is adapted whenever B_0 or k are changed. For each scan position in the subblock processing, the absolute level z is evaluated after encoding/decoding and B_0 is decremented by 1 after the first occurrence of $z > 1$, which corresponds to the first scan position in the subblock for which an ALG2 flag is signaled. A further adaptation of B_0 to its minimum value of 0 is performed after $z > 0$, i.e., after an ALG1 flag occurs eight times in the subblock. The parameter k is set to $\min(k + 1, 4)$ after each scan position for which the corresponding absolute level z fulfills the condition $z > 3 * 2^k$. Note that according to this adaptation rule, k can take integer values from 0 to 4 inclusive. Tables 8.12 and 8.13 show example binarizations for two different configurations of the parameters B_0, B_1, and k. Please note that the result of the binarization for z can also be interpreted as a concatenation of a unary prefix and, if present, a fixed-length suffix for different ranges of z [21]. Table 8.14 shows the corresponding binarization of ALRem, which has a maximum bin length of 32 [12].

As already indicated above, the signaling of the absolute level z involves four different syntax elements, given as `sig_coeff_flag` (SIG), `coeff_abs_level_greater1_flag` (ALG1), `coeff_abs_level_greater2_flag` (ALG2), and `coeff_abs_level_remaining` (ALRem), such that

$$z = \text{SIG} + \text{ALG1} + \text{ALG2} + \text{ALRem},$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00263

Table 8.13 Binarization of the absolute level z for the choice of parameters $B_0 = 1$, $B_1 = 9$, and $k = 1$, corresponding to a concatenation of TrU with cMax = 2, first-order Truncated Rice (TRk) with cMax = 8, and second-order Exp-Golomb (EGk)

		TrU		TRk					EGk			
		cMax=2		$k = 1$; cMax=8					$k + 1 = 2$			
z		SIG	ALG1	0	1	2	3	4	0	1	2	…
	0	0										
B_0 =	1	1	0									
	2	1	1	0	0							
	3	1	1	0	1							
	4	1	1	1	0	0						
	5	1	1	1	0	1						
	6	1	1	1	1	0	0					
	7	1	1	1	1	0	1					
	8	1	1	1	1	1	0	0				
B_1 =	9	1	1	1	1	1	0	1				
	10	1	1	1	1	1	1		0	0	0	
	11	1	1	1	1	1	1		0	0	1	
	…	…	…	…	…	…	…		…	…	…	…

Table 8.14 An alternative representation of `coeff_abs_level_remaining` (ALRem) binarization as a concatenation of a unary prefix and fixed length suffix

N	Prefix bins	Suffix bins	Prefix length	Suffix length	Total length	Max k
$0 \sim 2 \cdot 2^k - 1$	0	**C**	1	k	1+k	4
$1 \cdot 2^k \sim 2 \cdot 2^k - 1$	10	**C**	2	k	2+k	4
$2 \cdot 2^k \sim 3 \cdot 2^k - 1$	110	**C**	3	k	3+k	4
$2^k \cdot (2^0+2) \sim 2^k \cdot (2^1+2) - 1$	1110	**C**	4	k	4+k	4
$2^k \cdot (2^1+2) \sim 2^k \cdot (2^2+2) - 1$	11110	x**C**	5	1+k	6+k	4
$2^k \cdot (2^2+2) \sim 2^k \cdot (2^3+2) - 1$	111110	xx**C**	6	2+k	8+k	4
$2^k \cdot (2^3+2) \sim 2^k \cdot (2^4+2) - 1$	1111110	xxx**C**	7	3+k	10+k	4
$2^k \cdot (2^4+2) \sim 2^k \cdot (2^5+2) - 1$	11111110	xxxx**C**	8	4+k	12+k	4
$2^k \cdot (2^5+2) \sim 2^k \cdot (2^6+2) - 1$	111111110	xxxxx**C**	9	5+k	14+k	4
$2^k \cdot (2^6+2) \sim 2^k \cdot (2^7+2) - 1$	1111111110	xxxxxx**C**	10	6+k	16+k	4
$2^k \cdot (2^7+2) \sim 2^k \cdot (2^8+2) - 1$	11111111110	xxxxxxx**C**	11	7+k	18+k	4
$2^k \cdot (2^8+2) \sim 2^k \cdot (2^9+2) - 1$	111111111110	xxxxxxxx**C**	12	8+k	20+k	4
$2^k \cdot (2^9+2) \sim 2^k \cdot (2^{10}+2) - 1$	1111111111110	xxxxxxxxx**C**	13	9+k	22+k	4
$2^k \cdot (2^{10}+2) \sim 2^k \cdot (2^{11}+2) - 1$	11111111111110	xxxxxxxxxx**C**	14	10+k	24+k	4
$2^k \cdot (2^{11}+2) \sim 2^k \cdot (2^{12}+2) - 1$	111111111111110	xxxxxxxxxxx**C**	15	11+k	26+k	3
$2^k \cdot (2^{12}+2) \sim 2^k \cdot (2^{13}+2) - 1$	1111111111111110	xxxxxxxxxxxx**C**	16	12+k	28+k	2
$2^k \cdot (2^{13}+2) \sim 2^k \cdot (2^{14}+2) - 1$	11111111111111110	xxxxxxxxxxxxx**C**	17	13+k	30+k	1
$2^k \cdot (2^{14}+2) \sim 2^k \cdot (2^{15}+2) - 1$	111111111111111110	xxxxxxxxxxxxxx**C**	18	14+k	32+k	0

The suffix has a length of k bins when the value N of ALRem is less than $(3 << k)$; otherwise, it has a length of $\lfloor \log_2(((N - (3 << k)) >> k) + 1) \rfloor + k$ bins. In this table, the suffix bins are shown in terms of x and **C**, where each x represents a bin, and **C** represents a fixed length bin string of length k

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00264

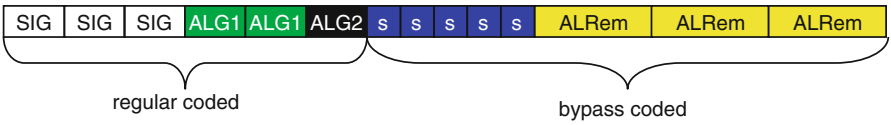


Fig. 8.23 Grouping same regular coded bins and bypass bins to increase throughput. s =coeff_sign_flag

provided that the values of the corresponding syntax elements are inferred to be equal to 0, when not explicitly signaled. Note that the flags SIG, ALG1, and ALG2 represent the first and the optional second and third bin indices of the TrU part of z, respectively. ALRem corresponds to the concatenation of the TRk and EGk part of z with all of its bin values being bypass coded and with a maximum bin string length of 32 [12]. Only the values of the three flags are regular coded. However, due to the adaptation rules for B_0, ALG2 can occur only once in each subblock, while the occurrence of ALG1 is restricted to 8 scan positions per subblock at the maximum [16]. Together with the maximum of 16 SIG flags per subblock, only up to 25 regular coded bins can occur in each subblock (without accounting for CSBF). Thus, the maximum number of regular coded bins per 4×4 transform (sub-) block is reduced by a factor of about 9.6 relative to the corresponding maximum number of $16 * 14 + 15 = 239$ regular coded bins for H.264/AVC CABAC (including SIG bins but without accounting for LAST) [46]. This change provides obviously the most substantial reduction to the (worst case) number of regular coded bins in the entire revision of CABAC.

The rationale behind processing SIG, ALG1, ALG2, and ALRem with individual syntax elements rather than as conventional bin indices of the adaptive binarization of z is given by the fact that all values of one syntax element in each subblock are grouped together and signaled in separate scan passes. This grouping provides essentially three advantages. First, bins in the coefficient level binarization that use the same context selection logic are grouped together to reduce the amount of speculative context selection computations, as shown in Fig. 8.23. Second, by grouping bypass coded bins together, the throughput advantages of bypass bins are maximized [87]. Third, the storage for (partially reconstructed) coefficient data during the parsing process at the decoder can be reduced, as further explained in Sect. 8.6.5.2 below. Note that the reordering of bins has no impact on coding efficiency.

Context modeling for coding of the regular coded bins of the absolute level is restricted to the three flags SIG, ALG1, and ALG2. Since context model selection for the SIG flag has already been introduced in Sects. 8.6.4.1 and 8.6.4.3, we will focus in the following on the two flags ALG1 and ALG2. For each of both flags, six sets of context models are provided: four sets for subblocks of the luma component and two sets for subblocks of the chroma component. Since only up to one ALG2 flag per subblock is encoded/decoded, each of the six ALG2 related sets contains only one context model. For the ALG1 flag, each set consists of four context models

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00265

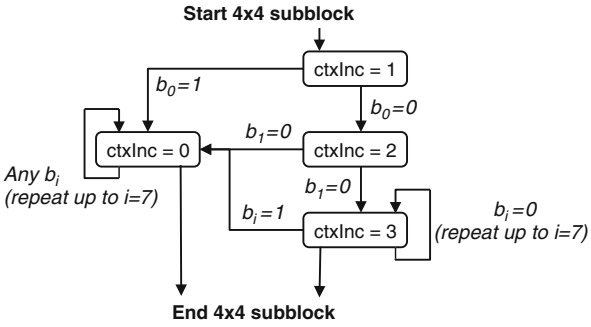


Fig. 8.24 Flow chart for derivation of context increment (ctxInc) for up to 8 different events b_i ($0 \leq i \leq 7$) of ALG1 in a 4×4 subblock

and the context increment ctxInc(ALG1) for selecting one of this four models within each set is quite similar to what is specified for the coding of the first bin of the syntax element `coeff_abs_level_minus1` in H.264/AVC (see [46] for a motivation of this design choice):

$$\text{ctxInc(ALG1)} = \begin{cases} 0, & \text{if NumG1} > 0 \\ 1 + \min(2, \text{NumT1}), & \text{otherwise} \end{cases},$$

where NumT1 denotes the accumulated number of encoded/decoded trailing 1's, i.e., absolute levels equal to 1, and NumG1 denotes the accumulated number of encoded/decoded levels with absolute value greater than 1, both computed along the reverse scanning pattern of the subblock up to (but not including) the current scan position. Note that both NumT1 and NumG1 are initialized with the value of 0 at the beginning of the subblock scan of ALG1 flags. After each encoded/decoded ALG1 flag with the value of 0, NumT1 is incremented by 1, while after each encoded/decoded ALG1 flag with the value of 1, NumG1 is incremented by 1. Figure 8.24 shows the flow chart for context increment computation of ALG1.

Since the statistics of trailing 1's may differ from subblock to subblock as well as for subblocks belonging to different components or different locations within the TB, different sets of context models are provided, both for ALG1 and ALG2, as already mentioned above. For subblocks belonging to the luma component, 2 separate sets are used for subblocks containing the DC of the TB, i.e., for the top left subblocks in a TB. Another two sets are given for luma subblocks containing no DC as well as two additional sets for chroma subblocks. Depending on the value of ctxInc(ALG1) for the last decoded ALG1 flag in the preceding subblock, the two members of each of the relevant sets related to luma DC, luma non-DC, and chroma are selected: One for the case of ctxInc(ALG1) $= 0$ and the other for the case of ctxInc(ALG1) > 0. Thus, a total number of 30 context models are used for coding of ALG1 and ALG2: $6 \cdot 4 = 24$ for ALG1 and 6 for ALG2. Interestingly enough, there was a $4\times$ reduction (from 120 to 30) in the total number of contexts used for coding of the ALG1 and ALG2 flags during the development from HM3.0 to HM6.0 at virtually no loss in coding efficiency.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00266

8.6.5.2 Coding of Sign

To reduce storage cost of the coefficients, as already noted above, the transform coefficient data is grouped for every 4 × 4 subblock and the sign bins are bypass coded and signaled before `coeff_abs_level_remaining` bins. Before `coeff_abs_level_remaining` is added, the partial value of the coefficient level can be represented with 4 bits. Thus, CABAC in HEVC only requires storage of $4 \times 4 \times 4$ bits for each subblock (as compared to $8 \times 8 \times 9$ bits for a 4 × 4 transform block in H.264/AVC), and the reconstructed transform coefficient level can be immediately written out once `coeff_abs_level_remaining` is parsed.

To improve coding efficiency, the optional *sign bit hiding* (SBH) technique can be used [24]. SBH is a technique to hide one bit such as, e.g., a sign of a non-zero coefficient in a group of non-zero coefficients. For this, the encoder quantizes the coefficients in the group such that the sum of their absolute level values is even or odd for the sign bit to be hidden having value 0 or 1, respectively. This inherently lossy coding technique is based on the idea that in a group of quantized coefficients, it is likely that there is at least one coefficient level for which the value can be increased or decreased by 1 with only marginally increased rate-distortion cost. This is, e.g., the case, when the unquantized coefficient was close to a quantization decision threshold, such that quantizing the coefficient to the next lower or next higher possible quantized value are both similarly good decisions.

SBH is enabled by `sign_data_hiding_enabled_flag` in the PPS and if it is enabled, it applies to each 4 × 4 subblock for which the number of non-zero coefficients exceeds a certain threshold. This threshold was chosen in HEVC to be a value of 3 and the sign bit to be hidden is that of the last significant scan position in the reverse scanning pattern of each subblock. The condition for SBH can be checked while parsing the significance map and thus, SBH does not have a significant impact on the entropy decoding throughput. Average bit rate savings between 0.6 and 0.9 % were reported for SBH at common test conditions [104].

8.6.5.3 Summary of Absolute Level and Sign Coding in HEVC

Figure 8.25 summarizes the last four out of up to five scan passes required for parsing the absolute levels and signs for every non-zero 4 × 4 subblock in HEVC.

8.6.6 Comparison of HEVC and H.264/AVC

Table 8.15 summarizes the differences in transform coefficient coding between HEVC and H.264/AVC as well as across different transform block sizes. In terms of throughput and memory related aspects, HEVC requires 3× fewer contexts (121 vs. 359) than H.264/AVC for transform coefficient coding. Note, however,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00267

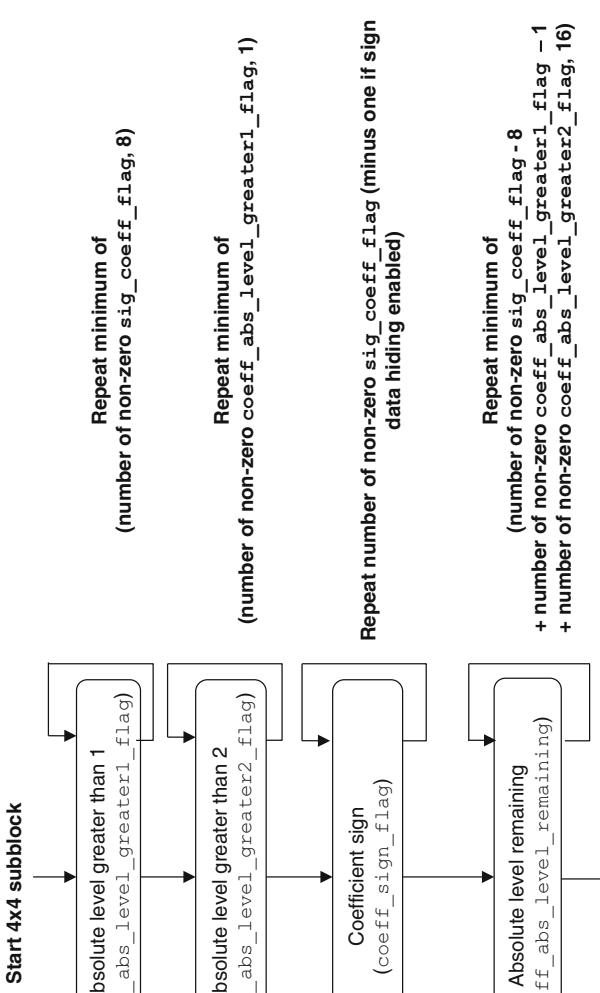


Fig. 8.25 Flow chart for coding the syntax elements of absolute level minus 1 and sign for a 4 × 4 subblock in HEVC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00268

that in H.264/AVC CABAC two separate sets of context models are used for frame-based and field-based coding of SIG and LAST. Furthermore, HEVC has a 9× lower maximum number of regular coded bins per coefficient (1.9 vs. 17.1) than H.264/AVC.

8.7 Context Initialization

In HEVC, slices consist of an integer number of CTUs, which collectively form an independently decodable unit. This implies in particular that at the beginning of each slice, the parameters of all probability models must be reset to some predefined values. Typically, without any prior knowledge of the statistical nature of the source, each probability model would be initialized with the state corresponding to the uniform distribution ($p = 0.5$). However, in order to bridge the learning phase of the adaptive probability models and to enable a kind of preadaptation at different coding conditions, it was found to be beneficial to provide some more appropriate initialization value than equi-probable state for each probability model at the beginning of each slice.

Similar to H.264/AVC, CABAC in HEVC involves a quantization-parameter dependent initialization process that is invoked at the beginning of each slice. It generates an initial probability state value representing the LPS probability p_{LPS} as well as the value of the MPS ν_{MPS} depending on the given initial value of the luma quantization parameter $\mathrm{SliceQP_Y}$ for the slice. For that purpose, a pair of so-called *initialization parameters* is stored for each model, from which a linear relationship between $\mathrm{SliceQP_Y}$ and the model probability p is derived. In contrast to H.264/AVC, the initialization parameters in HEVC do not directly represent the slope m and the offset n of the corresponding linear model. Instead, these two parameters are packed into a single 8 bit table entry in a memory-efficient way, as will be explained in more detail in the subsequent section.

For each of the three slice types I, P, and B, separate table entries are provided. However, for P and B slices the encoder can choose between the corresponding two table entries of initialization parameters and signal its choice to the decoder by use of the syntax element `cabac_init_flag`. Note that this mechanism is similar to that already available in H.264/AVC where, however, the choice between three instead of two pairs of initialization parameters is given for P and B slices [46, 63].

8.7.1 8-Bit Design

To reduce the memory requirements for context initialization tables, it was proposed in [52] to use 8-bit values to derive the initialization parameters rather than storing the pair of 16-bit values (m, n) of the linear model directly, as in H.264/AVC. From the high nibble of the 8-bit table entry InitValue, a variable slopeIdx is derived,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00269

Table 8.15 Differences between CABAC for different TB sizes in HEVC and H.264/AVC

Properties		**HEVC**				**H.264/AVC**	
TB Size		**4×4**	**8×8**	**16×16**	**32×32**	**4×4**	**8×8**
Context Selection for `sig_coeff_flag` (HEVC) or `significant_coeff_flag`		Position Based	Neighbor, Scan and Position Based	Neighbor and Position Based		Position Based	
Coefficient Scanning		Intra: Diagonal, Vertical, Horizontal Inter: Diagonal		Diagonal		Zig-zag	
Number of contexts	`coded_sub_block_flag`	2 (Y), 2 (CbCr)				n/a	n/a
	`sig_coeff_flag` (HEVC) or `significant_coeff_flag` (H.264/AVC)	8 (Y), 8 (CbCr)	12 (Y), 3 (CbCr)	6 (Y), 3 (CbCr)		41×2 (Y), 18×2 (CbCr)	15×2 (Y)
		1 (Y), 1 (CbCr) DC context shared across TU sizes					
	`last_sig_coeff_x,y` (HEVC) or `last_significant_coeff_flag` (H.264/AVC)	15×2 (Y), 3×2 (CbCr)				41×2 (Y) 18×2 (CbCr)	9×2 (Y)
	`coeff_abs_greater1_flag`, `coeff_abs_greater2_flag` (HEVC) or `coeff_abs_level_minus1` (H.264/AVC)	4×4 + 4×1 = 20 (Y), 2×4 + 2×1 = 10 (CbCr)				30 (Y), 19 (CbCr)	10 (Y)
Maximum regular coded bins per TB	`coded_sub_block_flag`	0	2	14	62	n/a	n/a
	`significant_coeff_flag`	15	63	255	1023	15	63
	`last_sig_coeff_x, y` (HEVC) or `last_significant_coeff_flag` (H.264/AVC)	3+3 = 6	5+5 = 10	7+7 = 14	9+9 = 18	15	63
	`coeff_abs_greater1_flag`, `coeff_abs_greater2_flag` (HEVC) or `coeff_abs_level_minus1` (H.264/AVC)	(8+1) ×1 = 9	(8+1) ×4 = 36	(8+1) ×16 = 144	(8+1) ×64 = 576	4×4 ×14 = 224	8×8 ×14 = 896
Maximum regular coded bins per coefficient		1.9	1.7	1.7	1.6	17.1	16.0

Number of contexts for H.264/AVC includes separate models for both SIG and LAST in frame and field coding mode (denoted by "×2" in the two rightmost columns for the corresponding syntax elements)

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00270

while the low nibble of InitValue represents the variable offsetIdx, from which the slope m and offset n of the linear model are derived using [29]

$$m = \text{slopeIdx} \cdot 5 - 45$$
$$n = (\text{offsetIdx} << 3) - 16.$$

Given the values of m and n, exactly the same initialization procedure as in H.264/AVC is performed for derivation of the parameters of each probability model [46,63]. Note that the 8-bit design allows to cut in half the amount of storage needed for context initialization tables. Further restriction to two instead of three table entries for P and B slice types reduces the memory requirements for those tables in HEVC by at least another 12.5–15 % relative to those of an 8-bit equivalent of H.264/AVC. Since there are 134 contexts for I slices and 154 for each of both slice types P and B, a total amount of 442 bytes of memory is needed for storage of all context initialization tables in HEVC.

8.7.2 Context Training

The main purpose of the context initialization tables is to bridge the learning phase starting from a uniform distribution, i.e., the case of no prior knowledge of the statistics of the given bin distributions, towards the well-adapted phase of the probability estimator. Assuming that after processing of a number of N_τ bins, the probability estimator that starts from $p = 0.5$ reaches such a well-adapted state, the bins for each probability model were tracked for N_τ bins for each test sequence of a training set at a particular QP and for a particular slice type. As a result, a model probability $p_{\tau,\text{QP}}$ was estimated from the relative frequency obtained after coding the first N_τ bins for each probability model. This training procedure was performed separately for each QP and each of the three slice types. To finally determine the pair of parameters (m, n) that describe the assumed linear relationship between QP and model probability $p_{\tau,\text{QP}}$, a simple linear regression was applied for each slice type. Note that a choice of $N_\tau = 50$ was assumed to be appropriate.

8.7.3 Context Memory for Wavefront Parallel Processing and Dependent Slices

For improving the parallelization and low-delay capabilities beyond the use of regular slices, as known from H.264/AVC, a partitioning of pictures into tiles, wavefronts and dependent slices have been introduced in HEVC. Since the use of regular slices implies in particular that the corresponding CABAC bitstream must be independently parsable, re-initialization of the CABAC probability models is required at the beginning of each regular slice. Although the initialization procedure,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00271

as described above, mitigates the effect of such a rigorous partitioning, the loss in coding efficiency is still too large to be acceptable for certain applications.

Wavefront parallel processing (WPP) is such a technique for picture partitioning with the focus on improving the capabilities for parallel processing at virtually no loss in coding efficiency [27, 34]. According to the WPP scheme, a picture is partitioned into rows of CTUs with each row being represented by its own CABAC bitstream which, however, is not fully independently parsable except for the bitstream belonging to first row of CTUs in a picture. Nevertheless, independent parsing and decoding of the WPP bitstreams is possible, if the processing from one CTU row to the next complies with an offset of two consecutive CTUs. This offset guarantees, on the one hand, that all spatial dependencies for the decoding process are preserved and, on the other hand, it permits inheritance of the adapted probability models from the first two CTUs in the preceding row of CTUs. The latter functionality, however, requires to store the content of all probability models after decoding the second CTU in a row. As already discussed above, the required memory depends on the slice type: for I slices 134 bytes and for P and B slices each 154 bytes of memory are needed. Note, however, that by using a proper scheduling and synchronization at the decoder, only one instance of such an additional context memory is required in addition to the N_ω context memories required for parsing and decoding N_ω CTU rows in parallel.

The same context memory handling applies also to the concept of dependent slice segments [69]. In HEVC, slices are composed of one initial independent slice segment and zero or more dependent slice segments, all of which contains an integer number of CTUs. Compared to regular slices or independent slice segments, dependent slice segments do not break the coding dependencies within the picture area to which the corresponding CTUs belong. Although each dependent slice segment has its own CABAC bitstream, the parsing of this bitstream cannot start before the parsing of the preceding dependent or independent slice segment has been finished. In particular, the content of all adapted probability models after parsing the last CTU in the preceding slice segment needs to be stored and propagated to the current dependent slice segment. Therefore, the same amount of additional context memory is required as in the WPP case. Note, however, that WPP and dependent slices, even though most often used together, are different concepts. While WPP is targeting at parallel processing, dependent slices cannot be processed in parallel and are most useful in applications requiring ultra-low delay, since each dependent slice segment can be put into a separate transport packet. Please refer to Chap. 3 for more details.

8.8 Overall Performance

This section analyzes the improvements of CABAC in HEVC relative to CABAC in H.264/AVC. In the first part of this section, the impact of all relevant CABAC changes in terms of coding efficiency is experimentally evaluated, while in the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00272

second part, an assessment of its throughput implications is performed. Finally, the reduction in memory requirements is analyzed.

Simulations were performed under common test conditions set by the JCT-VC [6, 101] as well as corresponding settings for H.264/AVC JM [30]. Note that those common conditions for the HEVC reference software HM [35] are intended to reflect the typical bitstreams in applications of HEVC. During standardization of HEVC, this configuration was also used to evaluate the coding efficiency impact of proposals.

In [6], four different test cases labeled as *Intra*, *Random Access*, *Low Delay B*, and *Low Delay P* are specified. The Intra test case specifies that all pictures are coded as intra pictures. In the Random Access test case, intra pictures are inserted in regular intervals of approximately 1.1 s in order to enable random access. As a temporal coding structure, hierarchical B pictures with groups of eight pictures are employed. Both the Low Delay B and Low Delay P test case specify that the pictures are coded in display order, so that the resulting structural encoding-decoding delay is suitable for low-delay communication applications. The latter two coding conditions differ only in the used slice type. In the Low Delay B test case, B slices are used, whereas only P slices are used in the Low Delay P test case. Note that in those low-delay test cases only one intra picture is used at the beginning of each test sequence.

The same set of test sequences as in the standardization process of HEVC has been used [6]. The test sequences are categorized into different classes, each with a particular spatial resolution. As an exception, the class labeled as *Screen content* in the following represents a special class that contains test sequences with typical screen and graphics content, but with varying spatial resolutions.

8.8.1 Coding Efficiency

Evaluation of coding efficiency for CABAC has been restricted to the syntax elements of transform coefficient coding. For that purpose, an extension of the residual coding scheme, specified for CABAC in H.264/AVC [46], was implemented into the HM to also cover residual coding of 16×16 and 32×32 TBs. This straightforward extension was realized by increasing the number of successive scan positions sharing the same context model for both SIG and LAST of those TBs. For the remaining syntax elements related to transform coefficient level coding, the same rules as defined for CABAC in H.264/AVC are applied [46].

Table 8.16 shows the so-called Bjøntegaard delta bit rate (BD-rate) for the luma component [5] as a measure of the gain in coding efficiency obtained for the transform coefficient level coding in HEVC relative to the aforementioned straightforward CABAC extension. Overall performance gains of 3.4–4.8 % in terms of averaged BD-rate savings can be attributed to the improved transform coefficient coding techniques in HEVC. The largest improvements are achieved for the Intra test case, which is mainly due to the relatively large energy of the corresponding residual signals.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00273

Table 8.16 BD-rate performance of CABAC transform coefficient coding in HEVC compared to the extended CABAC transform coefficient coding of H.264/AVC

Resolution and class of test sequences	Intra	Random Access	Low Delay B	Low Delay P
Class A: 2560 × 1600	−4.08	−2.86	n/a	n/a
Class B: 1920 × 1080	−4.18	−3.16	−3.17	−2.89
Class C: 832 × 416	−3.79	−2.82	−3.31	−3.13
Class D: 416 × 240	−4.15	−2.61	−2.43	−2.33
Class E: 1280 × 720	−4.92	n/a	−2.94	−2.69
Class F: Screen content	−7.74	−6.44	−5.79	−5.65
Average	**−4.78**	**−3.56**	**−3.54**	**−3.35**

Table 8.17 Coding efficiency impact of adopted TU coding tools

Tool	HM	Benefit	BD-rate
Neighbor based context selection for SIG [102]	1.0	coding gain	−2.8% to −1.4%
Group bypass sign [9]	1.0	throughput	0.0%
Mode dependent coefficient scanning [106]	2.0	coding gain	−1.2% to −0.1%
Reduce neighboring dependency for SIG [18]	2.0	throughput	−0.1% to 0.0%
Reduce regular coded level bins [54, 59]	3.0	throughput	−0.1% to 0.0%
Last position coding [73]	3.0	throughput	−0.1% to 0.0%
Group bypass level [87]	4.0	throughput	0.0%
Diagonal Scan [86]	4.0	throughput	−0.1% to 0.0%
CSBF & subblock scan [56, 76]	5.0	throughput	−0.1% to 0.1%
Reduce regular coded level bins per 4 × 4 [16]	6.0	throughput	−0.1% to 0.1%
Sign Bit Hiding [104]	6.0	coding gain	−0.9% to −0.6%
Use CSBF of neighboring subblocks for SIG [41]	7.0	throughput	0.1% to 0.2%

Note that positive BD-rate values indicate coding loss and negative BD-rate values indicate coding gain

Table 8.17 summarizes the individual coding efficiency impact of various adopted tools for HEVC. Note, however, that the majority of adopted tools focused on throughput improvements with minimal coding loss, as will be discussed in the following.

8.8.2 Throughput Analysis

This section describes throughput of HEVC relative to H.264/AVC. The impact of the techniques, outlined in Sect. 8.3.3, are discussed. Analysis was also done for the worst case throughput which is defined as the case with the maximum number of bins per 16×16 coding tree unit (CTU) or macroblock. The results for both common conditions and worst case are summarized in Tables 8.18 and 8.19, respectively.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00274

Table 8.18 Distribution of regular coded, bypass and termination bins for CABAC in H.264/AVC (JM-16.2) and HEVC (HM8.0) under common test conditions [6, 101]

	Common condition configurations	Context (%)	Bypass (%)	Term (%)
H.264/AVC	Hierarchical B	80.5	13.6	5.9
	Hierarchical P	79.4	12.2	8.4
HEVC	Intra	67.9	32.0	0.1
	Low Delay P	78.2	20.8	1.0
	Low Delay B	78.2	20.8	1.0
	Random Access	73.0	26.4	0.6

Table 8.19 Reduction of worst case number of bins and memory in HEVC over H.264/AVC

Metric	H.264/AVC	HEVC	Reduction
Max regular coded bins	7825	882	9×
Max bypass bins	13056	13417	1×
Max total bins	20882	14301	1.5×
Number of contexts	441	154	3×
Line buffer for 4k×2k	30720	1024	30×
Coefficient storage	8 × 8 × 9-bits	4 × 4 × 3-bits	12×
Initialization Table	1746 × 16-bits	442 × 8-bits	8×

Note max total bins includes termination mode bins, but does not include impact of bit limit per CTU or macroblock

8.8.2.1 Reduce Regular Coded Bins

As mentioned earlier, bypass coded bins can be processed faster than regular coded bins, since they don't have data dependencies due to context selection, and their interval subdivision can be performed by a simple shift. Table 8.18 shows that the percentage of regular coded bins under common conditions is lower for HEVC than H.264/AVC. Table 8.19 also shows that in the worst case conditions, there are 9× fewer regular coded bins in HEVC than H.264/AVC. The reduction in regular coded bins is primarily due to the improved binarizations of absolute coefficient levels and components of the motion vector difference.

Using the implementation found in [103], where up to 2 regular coded bins or 4 bypass coded bins can be processed per cycle, HEVC gives 2× higher throughput than H.264/AVC under the worst case (this includes the impact of 1.5× fewer total bins in HEVC). This can also be translated into power saving using voltage scaling as mentioned earlier.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00275

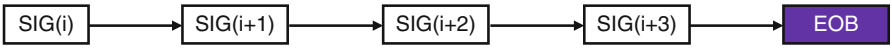


Fig. 8.26 No context speculation is required to achieve 5× parallelism when processing the 4 × 4 significance map in HEVC. i = coefficient position; EOB = end of block; SIG = `sig_coeff_flag`

8.8.2.2 Group Bypass Coded Bins

Grouping bypass bins together into longer chains increases the number of bins processed per cycle and reduces the number of cycles required to process a single bypass bin. This is a technique used in coding of syntax elements related to motion vector difference, intra mode, last position, and coefficient levels. For instance, for the Kimono sequence, encoded using the RandomAccess configuration, grouping bypass bins increases the average bypass bin run length from 2.1 to 6.4. In HEVC, under common test conditions, up to a 30 % reduction in number of cycles can be achieved compared to the case of no grouping [89].

The benefit of bypass grouping can also be seen in the example of Figs. 8.8 and 8.9. If bypass grouping was not used, it would take five cycles to process the 5 sign bypass bins. Assuming the architecture of [103], where 4 bypass bins are processed per cycle, only two cycles are required to process the 5 sign bins.

8.8.2.3 Group Bins with Same Context

Grouping bins with same context together is done for motion vector difference, significance map and coefficient level. As a results, fewer speculative calculations are needed to decode multiple bins per cycle since all bins that use the same logic for context selection are grouped together.

Figure 8.3 showed the speculation required when `significant_coeff_flag` and `last_significant_coeff_flag` are interleaved in H.264/AVC. In HEVC, no speculation is required for significance map as shown in Fig. 8.26. Thus for this example, the number of operations are reduced from 14 to 5.

8.8.2.4 Reduce Context Selection Dependencies

Context selection dependencies were reduced such that coding gains could be achieved without significant penalty to throughput. For instance, the last significant coefficient position information is sent before the SIG flag to remove a tight bin to bin data dependency. Relative to HM1.0, the neighboring dependencies for SIG were reduced from 10 to 5 neighboring SIG bins, and then further modified to only depend on neighboring 4 × 4 subblocks. The remaining context selection for SIG is only based on its position within the block as in H.264/AVC.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00276

Table 8.20 Summary of throughput improvement techniques with references to related standard contributions

Technique	PU coding	TU coding
Reduce regular coded bins	[58]	[16, 54, 59]
Group bypass bins	[19, 67]	[87]
Group bins with same context	[58]	[9, 10, 73]
Reduce context modeling dependencies		[18, 80, 85, 86]
Reduce total number of bins		[12, 56]
Reduce memory requirements	[58, 82, 90, 91]	[3, 15, 20, 62, 66, 81, 82, 93, 99]
Reduce parsing dependencies	[107, 108]	

8.8.2.5 Reduce Total Number of Bins

When comparing the total number of bins in the worst case, and thus the throughput requirement, HEVC has 1.5× fewer bins than H.264/AVC. Assuming the same number of cycles per bin are required, HEVC can run at a 1.5× lower clock rate at a lower voltage for 50 % power savings assuming linear scaling with voltage and frequency, or it can process at a bin rate that is 1.5× faster than H.264/AVC.

8.8.2.6 Reduce Parsing Dependencies

Parsing dependencies were removed or reduced such that coding gains could be achieve without significantly sacrificing throughput. Removing the parsing dependency for merge and mvp enables parsing to be mostly decoupled from the reconstruction process, as it is the case for H.264/AVC. HEVC does have parsing dependencies on intra mode reconstruction, which are not present in H.264/AVC; however, efforts were made to keep intra mode reconstruction simple to avoid affecting parsing throughput.

8.8.2.7 Summary of Throughput Improvement Techniques

Table 8.20 contains a summary of the techniques for throughput improvement and related standard contributions. An HEVC CABAC decoder that leverages several of these improvements to achieve a throughput of over 2 Gbin/s is described in [17].

8.8.3 Memory Requirement Reduction

This section describes how the size and bandwidth requirements of various memories in CABAC have been reduced in HEVC in order to increase throughput as well as lower implementation cost and power consumption.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00277

Table 8.21 Context memory requirements for H.264/AVC (4:2:0) and HEVC

	H.264/AVC		
	(w/ interlace)	(w/o interlace)	HEVC
CTU/CU contexts	25	22	16
PU contexts	26	26	14
TU contexts	390	244	124
Total	**441**	**292**	**154**

8.8.3.1 Context Memory

The motivation for context reduction was first proposed in [81], where the number of contexts was reduced for `coeff_abs_level_greater1_flag` and `coeff_abs_level_greater2_flag` without impacting coding efficiency. Subsequent proposals [3, 66, 93] were made to reduce the number of contexts for other syntax elements (e.g., `sig_coeff_flag`). HEVC uses only 154 contexts as compared to 441 (or 292 without interlaced) used in H.264/AVC as shown in Table 8.21; thus, a 3× reduction in context memory size is achieved with HEVC.

8.8.3.2 Line Buffer Memory

The motivation to reduce the size of the line buffer in the CABAC was first proposed in [90, 92], where the line buffer size was reduced by changing the context selection for motion vector difference. Subsequent proposals [15, 20, 58, 68, 82, 91] were made to further reduce neighboring dependencies to reduce the line buffer size. Based on these optimizations, in the worst case, the line buffer only need to store the CU depth (2-bits) of the top neighbor for context selection of `split_cu_flag` for every 8 × 8 block, and to indicate if the top neighbor is skipped (1-bit) for context selection of `cu_skip_flag` for ever 4 × 4 block. Assuming a minimum CU size of 8 × 8 for a 4k × 2k sequence, HEVC only requires a line buffer size of 1,024 bits versus 30,720 bits in H.264/AVC, which is a 30 × reduction.

8.8.3.3 Coefficient Storage

Large TB sizes have large hardware cost implications. Compared to H.264/AVC, the 16 × 16 and 32 × 32 TBs in HEVC have 4× and 16× more coefficients than an 8 × 8 TB, respectively, and consequently require an increase in storage cost. Several techniques were used to reduce the coefficient storage cost. First, the sign information is sent before `coeff_abs_level_remaining` such that only 3-bits storage is required per coefficient for the partial decoded value (if stored as a 2-bit number with a range from 0 to 3, and a sign bit). Second, the coefficient information is interleaved at a 4 × 4 subblock level, such that the fully constructed coefficient can be achieved for every subblock and be sent out to the next module [75]. Thus, only a coefficient storage of 4 × 4 × 3-bits is required in HEVC CABAC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00278

(compared with 8 × 8 × 9-bit in H.264/AVC) in order to reconstruct the coefficient levels.

8.8.3.4 Context Initialization Tables

As already discussed in Sect. 8.7, the memory requirements for storing the context initialization tables in HEVC have been reduced to a large extent when compared to those of H.264/AVC. Accounting for the reduction in number of contexts, number of bits per InitValue and number of InitValue sets, HEVC has an 9× smaller context initialization table than H.264/AVC.

8.9 Conclusions

Entropy coding was a highly active area of development throughout the HEVC standardization process with proposals for both coding efficiency and throughput improvement. The trade-off between the two requirements were carefully evaluated in multiple Core Experiments and Ad Hoc Groups [8, 11, 13, 37, 97]. Beside coding-efficiency improving technology, many techniques were incorporated to improve throughput including reducing regular coded bins, grouping bypass bins together, grouping bins that use the same contexts together, reducing context selection dependencies, and reducing the total number of signaled bins. CABAC memory requirements were also significantly reduced. The final design of CABAC in HEVC shows that by accounting for implementation cost *and* coding efficiency when designing entropy coding algorithms results in a design that can maximize processing speed and minimize area cost, while delivering high coding efficiency in the latest video coding standard.

References

1. Alshina E, Alshin A (2011) Multi-parameter probability up-date for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F254, Torino, July 2011
2. Amonou I, Cammas N, Clare G, Jung J, Noblet L, Pateux S, Matsuo S, Takamura S, Boon CS, Bossen F, Fujibayashi S, Kanumuri S, Suzuki Y, Takiue J, Tan TK, Drugeon V, Lim CS, Narroschke M, Nishi T, Sasai H, Shibahara Y, Uchibayashi K, Wedi T, Wittmann S, Bordes P, Gomila C, Guillotel P, Guo L, François E, Lu X, Sole J, Vieron J, Xu Q, Yin P, Zheng Y (2010) Video coding technology proposal by France Telecom, NTT, NTT DoCoMo, Panasonic and Technicolor, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A114, Dresden, Apr. 2010
3. Auyeung C, Xu J, Korodi G, Zan J, He D, Piao Y, Alshina E, Min J, Park J (2011) A combined proposal from JCTVC-G366, JCTVC-G657, and JCTVC-G768 on context reduction of significance map coding with CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1015, Geneva, Nov. 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00279

4. Belyaev E, Gilmutdinov M, Turlikov A (2006) Binary arithmetic coding system with adaptive probability estimation by "virtual sliding window". In: 2006 IEEE tenth international symposium on consumer electronics (ISCE '06), pp 1–5
5. Bjøntegaard G (2001) Calculation of average PSNR differences between RD curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001
6. Bossen F (2012) HM 8 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J1100, Stockholm, July 2012
7. Bross B, Jung J (2011) Description of core experiment CE13: motion data parsing robustness and throughput, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F913, Torino, July 2011
8. Budagavi M (2010) Tool experiment 8: parallel entropy coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-B308, Geneva, July 2010
9. Budagavi M (2010) TE8: TI parallel context processing (PCP) proposal, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C062, Guangzhou, Oct. 2010
10. Budagavi M, Demircin MU (2010) Parallel context processing techniques for high coding efficiency entropy coding in HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-B088, Geneva, July 2010
11. Budagavi M, Martin-Cocher G, Segall A (2010) JCT-VC AHG report: entropy coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D009, Daegu, Jan. 2010
12. Budagavi M, Sze V (2012) coeff_abs_level_remaining maximum codeword length reduction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0142, Stockholm, July 2012
13. Budagavi M, Segall A (2010) AHG report: parallel entropy coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-B009, Geneva, July 2010
14. Chandrakasan A, Sheng S, Brodersen R (1992) Low-power CMOS digital design. IEEE J Solid-State Circuits 27(4):473–484. doi:10.1109/4.126534
15. Chen C, Lee T (2011) Simplified context model selection for block level syntax coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F497, Torino, July 2011
16. Chen J, Chien WJ, Joshi R, Sole J, Karczewicz M (2012) Non-CE1: throughput improvement on CABAC coefficients level coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0554, San Jose, Feb. 2012
17. Y. H. Chen, V. Sze, "A 2014 Mbin/s Deeply Pipelined CABAC Decoder For HEVC," IEEE International Conference on Image Processing (ICIP), Oct. 2014
18. Cheung A, Lui W (2011) Parallel processing friendly simplified context selection of significance map, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D260, Daegu, Jan. 2010
19. Chien WJ, Chen J, Coban M, Karczewicz M (2012) Intra mode coding for INTRA_NxN, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0302, Geneva, Apr. 2012
20. Chien WJ, Karczewicz M, Wang X (2011) Memory and parsing friendly CABAC context, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F606, Torino, July 2011
21. ChienWJ, Karczewicz M, Sole J, Chen J (2012) On coefficient level remaining coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0487, Geneva, Apr. 2012
22. Chono K (2012) BoG report on intra mode coding cleanup and simplification, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0712, San Jose, Feb. 2012
23. Chono K, Aoki H (2011) Efficient binary representation of cu_qp_delta syntax for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F046, Torino, July 2011
24. Clare G, Henry F, Jung J (2011) Sign data hiding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G271, Geneva, Nov. 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00280

25. Finchelstein D, Sze V, Chandrakasan A (2009) Multicore processing and efficient on-chip caching for H.264 and future video decoders. IEEE Trans CSVT 19(11):1704–1713
26. Fuldseth A, Bjøntegaard G, Budagavi M, Sze V (2011) CE10: core transform design for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G495, Geneva, Nov. 2011
27. Gordon C, Henry F, Pateux S (2011) Wavefront parallel processing for HEVC encoding and decoding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F274, Torino, July 2011
28. Guo X, Huang YW, Lei S (2009) Ordered entropy slices for parallel CABAC, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-AK25, Yokohoma, Apr. 2009
29. Guo L, Sole J, Joshi R, Karczewicz M, Yeo C, Tan Y, Li Z (2012) CE1 B3: 8-bit linear initialization for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0535, San Jose, Feb. 2012
30. H.264/AVC Reference Software, JM 16.2. http://iphome.hhi.de/suehring/tml/, 2009
31. He D, Korodi G, Martin-Cocher G, Yang EH, Yu X, Zan J (2010) Video coding technology proposal by research in motion, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A120, Dresden, Apr. 2010
32. Helle P, Oudin S, Bross B, Marpe D, Bici M, Ugur K, Jung J, Clare G, Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. IEEE Trans CSVT 22(12): 1720–1731
33. Hellman T, Yu Y (2011) Decoder performance restrictions due to merge/MVP index parsing, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F341, Torino, July 2011
34. Henry F, Pateux S (2011) Wavefront parallel processing, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196, Geneva, Mar. 2011
35. HEVC Test Model, HM 8.0. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-8.0/, 2012
36. Huang YW, Alshina E (2012) BoG report on integrated text of SAO adoptions on top of JCTVC-I0030, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0602, Geneva, Apr. 2012
37. Joshi R, Alshina E, Sasai H, Kirchhoffer H, Lainema J (2011) Description of core experiment 1: entropy coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F901, Torino, July 2011
38. Jung J, Laroche G (2006) Competition-based scheme for motion vector selection and coding, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-AC06, Klagenfurt, July 2006
39. Karczewicz M, Chen P, Joshi R, Wang X, Chien WJ, Panchal R (2010) Video coding technology proposal by Qualcomm, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A121, Dresden, Apr. 2001
40. Kim WS, Kwon DK (2011) Non-CE8: method of visual coding artifact removal for SAO, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G0680, Geneva, Mar. 2011
41. Kumakura T, Fukushima S (2012) Non-CE3: simplified context derivation for significance map, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0296, Geneva, Apr. 2012
42. Lan C, Xu J, Sullivan GJ, Wu F (2012) Intra transform skipping, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0408, Geneva, Apr. 2012
43. Marpe D, Wiegand T (2003) A highly efficient multiplication-free binary arithmetic coder and its application in video coding. In: IEEE international conference on image processing, pp 263–266
44. Marpe D, Blättermann G, Wiegand T (2001) Adaptive codes for H.26L, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-L13, Eibsee, Jan. 2001
45. Marpe D, Heising G, Blättermann G,Wiegand T (2002) Fast arithmetic coding for CABAC, Joint Video Team (JVT), Document JVT-C061, Fairfax, Mar. 2002

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00281

46. Marpe D, Schwarz H, Wiegand T (2003) Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans CSVT 13(7):620–636
47. Marpe D, Marten G, Cycon HL (2006) A fast renormalization technique for H. 264/MPEG4-AVC arithmetic coding. In: 51st Internationales Wissenschaftliches Kolloquium Technische Universität Ilmenau
48. Marpe D, Marten G, Wiegand T (2006) Fast CABAC renormalization for H.264/MPEG4-AVC. Joint Video Team (JVT), Document JVT-U084, Hangzhou, Oct. 2005
49. Marpe D, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H, Oudin S, Schwarz H, Siekmann M, Sühring K, Winken M, Wiegand T (2010) Video compression using nested quadtree structures, leaf merging and improved techniques for motion representation and entropy coding. IEEE Trans CSVT 20(12):1676–1687
50. Marpe D, Schwarz H, Wiegand T (2010) Entropy coding in video compression using probability interval partitioning. In: Picture coding symposium (PCS), pp 66–69
51. Marpe D, Schwarz H, Wiegand T (2010) Novel entropy coding concept, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A032, Dresden, Apr. 2010
52. Marpe D, Kirchhoffer H, Bross B, George V, Nguyen T, PreißM, Siekmann M, Stegemann J, Wiegand T (2011) Unified PIPE-based entropy coding for HEVC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F268, Torino, July 2011
53. McCann K, Bross B, Sekiguchi S, HanWJ (2010) HEVC test model 1 (HM 1) encoder description, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C402, Guangzhou, Oct. 2010
54. Nguyen T (2011) CE11: coding of transform coefficient levels with Golomb-Rice codes, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E253, Geneva, Mar. 2011
55. Nguyen T, Schwarz H, Kirchhoffer H, Marpe D, Wiegand T (2010) Improved context modeling for coding quantized transform coefficients in video compression. In: Picture coding symposium (PCS), pp 378–381
56. Nguyen N, Ji T, He D, Martin-Cocher G, Song L (2011) Multi-level significant maps for large transform units, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G644, Geneva, Nov. 2011
57. Nguyen T, Marpe D, Schwarz H, Wiegand T (2011) CE11: evaluation of transform coding tools in HE configuration, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D061, Daegu, Jan. 2011
58. Nguyen T, Marpe D, Schwarz H, Wiegand T (2011) Modified binarization and coding of MVD for PIPE/CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F455, Torino, July 2011
59. Nguyen T, Winken M, Marpe D, Schwarz H, Wiegand T (2011) Reduced complexity entropy coding of transform coefficient levels using a combination of VLC and PIPE, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D336, Daegu, Jan. 2011
60. Nguyen N, Ji T, He D, Martin-Cocher G (2012) Non-CE1: throughput improvement on CABAC coefficients level coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0554, San Jose, Feb. 2012
61. Peng X, Lan C, Xu J, Sullivan GJ (2012) Inter transform skipping, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0237, Stockholm, July 2012
62. Piao Y, Min J, Alshina E, Park JT (2011) Reduced chroma contexts for significance map coding in CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G781, Geneva, Nov. 2011
63. ITU-T Rec. H.264 and ISO/IEC 14496-10 (2003) Advanced video coding
64. ITU-T Rec. H.265 and 23008-2 (2013) High efficiency video coding
65. Ryabko BY (1996) Imaginary sliding window as a tool for data compression. Probl Inf Transm 32(2):156–163
66. Sasai H, Nishi T (2011) CE11: context size reduction for the significance map, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E227, Geneva, Mar. 2011
67. Sasai H, Nishi T (2011) Modified MVD coding for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F423, Torino, July 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00282

68. Sasai H, Nishi T (2011) Modified context derivation for neighboring dependency reduction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F429, Torino, July 2011
69. Schierl T, Goerge V, Henkel A, Marpe D (2012) Dependent slices, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0229, Geneva, Apr. 2012
70. Seregin V, Kim IK (2011) Binarisation modification for last position coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F375, Torino, July 2011
71. Seregin V, Sole J, Karczewicz M,Wang X, Sze V, Budagavi M (2012) AHG5: bypass bins for reference index coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0098, Stockholm, July 2012
72. Shannon CE (1948) A mathematical theory of communications. Bell Syst Tech J 27:379–423
73. Sole J, Joshi R, Karczewicz M (2011) CE11: parallel context processing for the significance map in high coding efficiency, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E338, Geneva, Mar. 2011
74. Sole J, Joshi R, Karczewicz M (2011) CE11: unified scans for the significance map and coefficient level coding in high efficiency, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F288, Torino, July 2011
75. Sole J, Joshi R, Karczewicz M (2011) CE11: scanning of residual data in HE, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F552, Torino, July 2011
76. Sole J, Joshi R, Karczewicz M (2011) Non-CE11: diagonal sub-block scan for HE residual coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G323, Geneva, Nov. 2011
77. Sole J, Joshi R, Karczewicz M (2012) Removal of the 8x2/2x8 coefficient groups, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0256, Stockholm, July 2012
78. Stegemann J, Kirchhoffer H, Marpe D, Wiegand T (2011) Non-CE1: counterbased probability model update with adapted arithmetic coding engine, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G547, Geneva, Nov. 2011
79. Sugio T, Nishi T (2011) Parsing robustness for Merge/AMVP, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F470, Torino, July 2011
80. Sze V (2011) Context selection complexity in HEVC CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D244, Daegu, Jan. 2011
81. Sze V (2011) Reduction in contexts used for significant_coeff_flag and coefficient level, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F132, Torino, July 2011
82. Sze V (2011) BoG report on context reduction for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F746, Torino, July 2011
83. Sze V, Allen R (2011) BoG report on intra mode coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1017, Geneva, Nov. 2011
84. Sze V, Budagavi M (2008) Parallel CABAC, ITU-T SG16 Q6, Document COM-16-C-334-E, Geneva, Apr. 2008
85. Sze V, Budagavi M (2010) Parallelization of HHI_TRANSFORM_CODING, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C227, Guangzhou, Oct. 2010
86. Sze V, Budagavi M (2011) CE11: parallelization of HHI_TRANSFORM_CODING fixed diagonal scan, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F129, Torino, July 2011
87. Sze V, BudagaviM(2011) Parallel context processing of coefficient level, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F130, Torino, July 2011
88. Sze V, Budagavi M (2012) High throughput CABAC entropy coding in HEVC. IEEE Trans CSVT 22(12):1778–1791. doi:10.1109/TCSVT.2012.2221526
89. Sze V, Budagavi M (2013) A comparison of CABAC throughput for HEVC/H.265 vs. AVC/H.264. In: IEEE workshop on signal processing systems
90. Sze V, Chandrakasan AP (2011) Joint algorithm-architecture optimization of CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E324, Geneva, Mar. 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00283

91. Sze V, Chandrakasan AP (2011) Simplified MVD context selection (extension of JCTVC-E324), Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F133, Torino, July 2011
92. Sze V, Chandrakasan AP (2011) Joint algorithm-architecture optimization of CABAC to increase speed and reduce area cost. In: IEEE international conference on acoustics, speech and signal processing, pp 1577–1580
93. Sze V, Sasai H (2011) Modification to JCTVC-E227 in CE11 for reduced dependency with MDCS, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E489, Geneva, Mar. 2011
94. Sze V, Budagavi M, Chandrakasan A, Zhou M (2008) Parallel CABAC for low power video coding. In: IEEE international conference on image processing, pp 2096–2099
95. Sze V, Budagavi M, Demircin MU (2008) CABAC throughput requirements for real-time decoding, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-AJ31, San Diego, Oct. 2008
96. Sze V, Budagavi M, Chandrakasan A (2009) Massively parallel CABAC, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-AL21, Geneva, July 2009
97. Sze V, Panusopone K, Chen J, Nguyen T, Coban M (2010) Description of core experiment 11: coefficient scanning and coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C511, Guangzhou, Oct. 2010
98. Sze V, Budagavi M, Seregin V, Sole J, Karczewicz M (2012) AHG5: bin reduction for delta QP coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0089, Stockholm, July 2012
99. Terada K, Sasai H, Nishi T (2012) Non-CE11: simplification of context selection for significant_coeff_flag, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0290, San Jose, Feb. 2012
100. Ugur K, Saxena A (2012) CE1: summary report of core experiment on intra transform mode dependency simplifications, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0021, Stockholm, July 2012
101. ITU-T SG16 Q6 and ISO/IEC JTC1/SC29/WG11 (2010) Joint call for proposals on video compression technology. ITU-T SG16 Q6 document VCEG-AM91 and ISO/IEC JTC1/SC29/WG11 document N11113, Kyoto, 22 Jan. 2010
102. Winken M, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H, Marpe D, Oudin S, PreißM, Schwarz H, Siekmann M, Sühring K, Wiegand T (2010) Description of video coding technology proposal by Fraunhofer HHI, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-A116, Dresden, Apr. 2010
103. Yang YC, Guo JI (2009) High-throughput H.264/AVC high-profile CABAC decoder for HDTV applications. IEEE Trans CSVT 19(9):1395–1399. doi:10.1109/TCSVT.2009.2020340
104. Yu X,Wang J, He D, Martin-Cocher G, Campbell S (2012) Multiple sign bits hiding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0481, San Jose, 2012
105. Zhao J, Segall A (2008) Entropy slices for parallel entropy decoding, ITU-T SG16 Q6, Document COM-16-C-405-E, Geneva, Apr. 2008
106. Zheng Y, Coban M, Wang X, Sole J, Joshi R, Karczewicz M (2011) CE11: mode dependent coefficient scanning, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D393, Daegu, Jan. 2011
107. Zhou M, Sze V (2011) A study on HM2.0 bitstream parsing and error resiliency issue, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E118, Geneva, Mar. 2011
108. Zhou M, Sze V, Mastuba Y (2011) A study on HEVC parsing throughput issue, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F068, Torino, July 2011

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00284

Chapter 9
Compression Performance Analysis in HEVC

Ali Tabatabai, Teruhiko Suzuki, Philippe Hanhart, Pavel Korshunov, Touradj Ebrahimi, Michael Horowitz, Faouzi Kossentini, and Hassene Tmar

Abstract In this chapter, performance analysis of HEVC (Recommendation ITU-T H.265 | ISO/IEC 23008-2) in comparison with AVC (Recommendation ITU-T H.264 | ISO/IEC 14996-10) in terms of both objective as well as subjective quality assessments are given. Because of the increased flexibility offered by HEVC, methods to select the best coding parameters, in a rate-distortion sense, are also described. Special care has been taken to apply a unified approach when conducting subjective and objective quality evaluation between HEVC and AVC. Our overall evaluation study results show the coding efficiency of HEVC to be about twice higher than that of AVC.

9.1 Performance Analysis

Performance analysis of HEVC is in general a complex undertaking since it can be conducted in number of different ways based on, for example, compression efficiency, complexity, visual quality, application of rate distortion optimization (RDO), delay, robustness, etc. The goal of this chapter is to present HEVC compression efficiency in comparison with AVC both in terms of objective and subjective quality assessments while taking into account some aspects of complexity, RDO,

A. Tabatabai • T. Suzuki (✉)
Sony Corporation, Tokyo, Japan
e-mail: ali.tabatabai@am.sony.com; teruhikos@jp.sony.com

P. Hanhart • P. Korshunov • T. Ebrahimi
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

M. Horowitz • F. Kossentini • H. Tmar
eBrisk Video, Inc., Vancouver, BC, Canada

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_9
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00285

and delay. Note that it is important to include both quality measures; relying solely on objective quality evaluations could, in cases, underestimate the amount of bit rate reduction and hence affect our analysis of compression efficiency. Subjective quality evaluations on the other hand, although difficult to conduct, correlate directly with perceptual experience of the viewers. This chapter is organized as follows. Section 9.1 provides the background information and forms the basis for the sections that will follow. In Sect. 9.2 encoder settings and testing conditions are described by considering various encoder configurations according to complexity and delay requirements; moreover, a list of test sequences used, test cases and the description of non-normative R–D optimization tools that contribute significantly to coding efficiency improvement are also covered in this section. In Sect. 9.3, objective quality evaluations of HEVC and AVC reference implementations are investigated. In Sect. 9.4, we present the results of HEVC subjective quality testing and visual assessments of HEVC and AVC. Section 9.5 describes an informal subjective video quality comparison of production–quality HEVC and AVC encoders in the context of 4K streaming applications. Conclusions appear in Sect. 9.6.

9.2 Encoder Setting

To conduct HEVC and AVC performance evaluations, a well-defined encoder setting and testing environment need to be established. In this section, we will describe, the HEVC and AVC reference encoder software (SW) used in our investigations. In addition, we will also describe various encoder configurations and prediction structures that are appropriate for different application requirements in terms of coding efficiency, complexity and delay.

9.2.1 Encoder Software

In the standardization of HEVC, the reference software, which is called HM (HEVC Test Model, reference software) [15] has been developed as a common SW platform for further improvement and study. Using SVN servers, the HM reference software is maintained at two sites [16]: HHI (Heinrich Hertz Institute) maintains the main SVN server and BBC (British Broadcasting Corporation) maintains the mirroring repository site.

The reference software for AVC, which is called JM (Joint Test Model), has been developed, as a common test platform, for AVC performance evaluations. The JM reference software is maintained at SVN server [6]. In this chapter, in order to compare the coding performance of HEVC with AVC, HM12.1 and JM18.5 SW are used for HEVC and AVC encoders, respectively.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00286

9.2.2 Test Conditions

During the development of the HEVC specification, establishment of Common Test Conditions (CTC) provided a well-defined platform on which experiments for coding tool evaluations are performed [3]. Since HEVC coding performance evaluations are carried out according to the CTC, a detailed description of CTC key elements will follow.

9.2.3 Prediction Structure

For performance evaluation, CTC defines the following prediction structures.

1. All Intra (AI)
2. Random Access (RA)
3. Low Delay P picture (LDP)
4. Low Delay B picture (LDB)

In these configurations, QP (Quantization Parameter) value can be modified by adding to it a "QP offset" value. That is, CTC defines QP of the first picture (QP of an I picture, QPI, with I picture defined below) and the QP of the following pictures are derived as QP = (QPI + QP offset), with QP offset being determined according to the picture type (e.g., P & B pictures, defined below) or a picture temporal ID. An I (intra) picture refers to a picture that can be decoded independently without requiring prediction data from other decoded pictures. A P (predicted) picture, in general, requires picture sample data from one other I, P or B picture to generate each predicted sample block. A B (bi-predicted), in general, requires picture sample data from two other I, P or B pictures to generate each predicted sample block.

9.2.3.1 All Intra (AI)

In this configuration, each picture is encoded as an I picture. Because no inter picture prediction is used, it is thus suitable for low delay and higher bit rate applications. QP offset in this configuration is 0 since QP is kept constant over the whole sequence. Figure 9.1 shows an example of this prediction structure.

9.2.3.2 Random Access (RA)

In this configuration, a hierarchical B structure is used [21]. Figure 9.2 shows an example of this prediction structure. The coding efficiency achieved by the bi-directional hierarchical prediction structure is higher than the other configurations. It has however a larger delay due to the reordering of the pictures. To control possible error propagation and ease of random access, I pictures are inserted periodically. QP offset values for each picture are summarized in Fig. 9.2.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00287

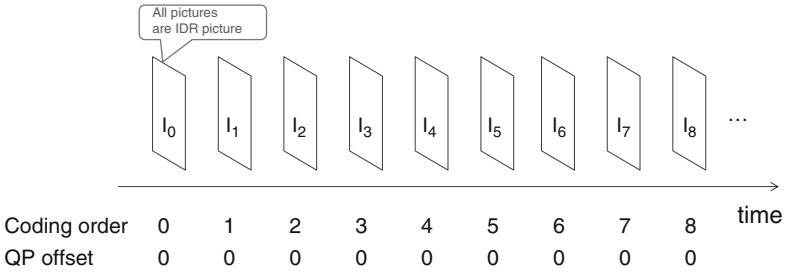


Fig. 9.1 The prediction structure of the intra-only configuration

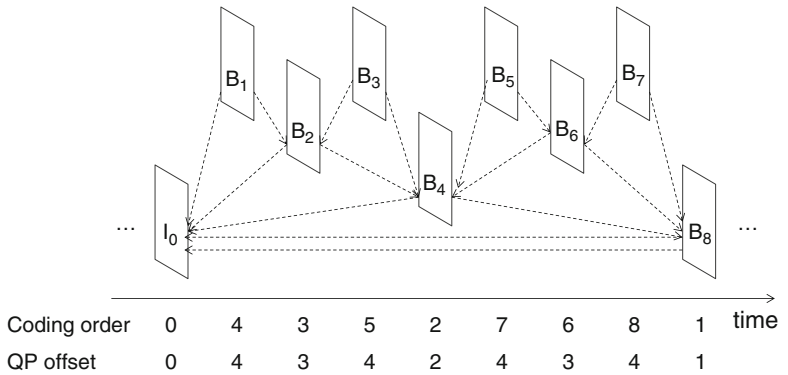


Fig. 9.2 The prediction structure of the random access configuration

9.2.3.3 Low-Delay P (LDP)

In this configuration, the first picture is encoded as an I picture and the subsequent pictures are encoded as P pictures. Since reordering of pictures is not allowed and only past pictures are used for prediction, the coding delay, in this configuration, may be made small. Figure 9.3 shows an example of this prediction structure. QP offset values are summarized for each picture in Fig. 9.3.

9.2.3.4 Low-Delay B (LDB)

In this configuration, similar to the previous configuration, reordering of pictures is not allowed. The first picture is encoded as an I picture and subsequent pictures are encoded as B pictures. Moreover, since past B pictures are used for prediction, a low coding delay, similar to LDP, but with higher coding efficiency (because of bi-prediction) is achieved. QP offset values, for each picture, are summarized in Fig. 9.3.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00288

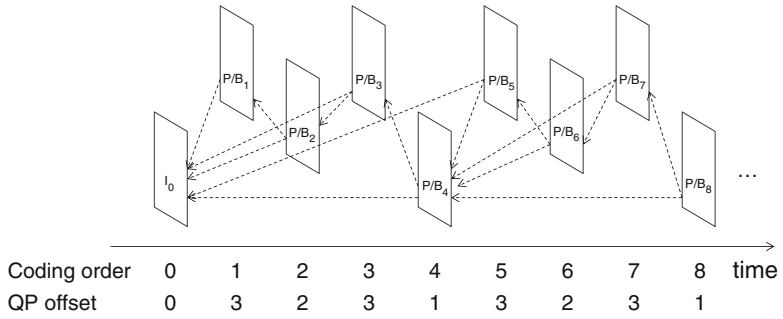


Fig. 9.3 The prediction structure of low-delay P and B configurations

9.2.4 Test Sequences

Test sequences are defined according to the picture size and applications and they are classified into six classes (class A to class F). Class A is the set of sequences with higher resolution than 1080p HDTV. The sequences are used to evaluate the coding performance of 4K/8K video. To reduce computation time, picture sizes are cropped to $2{,}560 \times 1{,}600$ pixels. Class B is for coding performance evaluation of 1080p HDTV and the set contains HDTV sequences, with a picture size of $1{,}920 \times 1{,}080$ pixels. Classes C and D are the set of test sequences with picture sizes of 832×480 pixels and 416×240 pixels, respectively. Test sequences in these two classes are for coding performance evaluation of mobile applications. Class E is the set of test sequences with a picture size of $1{,}280 \times 720$ pixels. It is used to evaluate coding performance of low-latency applications such as visual communications. CTC, in addition, defines class F sequences for coding performance evaluation of non-camera captured content such as video screen content, containing, for example, text and computer graphic. The test sequences are listed in Table 9.1.

In addition to the test sequences defined in CTC, 4K test sequences listed in Table 9.2 are used for both objective and subjective quality performance analysis in this chapter.

9.2.5 Test Cases and Bit Depth

Two test cases Main and Main 10 are defined to evaluate coding performance of 8-bit and 10-bit video. All test cases are summarized in Table 9.3.

In Main10 configuration, an 8-bit video is converted first to a 10-bit video by a 2-bit left shift, and it is then encoded as 10-bit video. Likewise, in Main configuration, a 10-bit video is first converted to an 8-bit video by a 2-bit right shift,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00289

Table 9.1 Test sequences

Class	Sequence name	Frame count	Frame rate (fps)	Bit depth
A	Traffic	150	30	8
A	PeopleOnStreet	150	30	8
A	Nebuta	300	60	10
A	SteamLocomotive	300	60	10
B	Kimono	240	24	8
B	ParkScene	240	24	8
B	Cactus	500	50	8
B	BQTerrace	600	60	8
B	BasketballDrive	500	50	8
C	RaceHorses	300	30	8
C	BQMall	600	60	8
C	PartyScene	500	50	8
C	BasketballDrill	500	50	8
D	RaceHorses	300	30	8
D	BQSquare	600	60	8
D	BlowingBubbles	500	50	8
D	BasketballPass	500	50	8
E	FourPeople	600	60	8
E	Johnny	600	60	8
E	KristenAndSara	600	60	8
F	BaskeballDrillText	500	50	8
F	ChinaSpeed	500	30	8
F	SlideEditing	300	30	8
F	SlideShow	500	20	8

Table 9.2 4K Test sequences

Resolution	Sequence name	Frame count	Frame rate (fps)	Bit depth
3,840 × 2,160	Book[a]	500	50	10
3,840 × 2,160	BT709Birthday[b]	500	50	10
3,840 × 2,160	HomelessSleeping[c]	600	60	10
3,840 × 2,160	Manege[d]	600	60	8
4,096 × 2,048	Traffic	300	30	8

[a]Book sequence was created by BBC (British Broadcasting Corporation), Research and Development Department
[b]BT709Birthday sequence was created by Technicolor
[c]HomelessSleeping sequence was created by Kamerawerk GmbH, Switzerland. The sequence is an excerpt from the film entitled "No Sleep 4K"
[d]Manege sequence was created by 4EVER consortium

and it is then encoded as an 8-bit video. The word "optional" in Table 9.3 means that using certain class of sequences (e.g., class F) or certain prediction structures (e.g., LDP) were not required but recommended, instead. In this chapter, Main configuration is used to evaluate "optional" cases in Table 9.3.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00290

Table 9.3 Summary of test cases in the common test conditions

Prediction structure	Class					
	A	B	C	D	E	F
AI	Main/Main10	Main/Main10	Main/Main10	Main/Main10	Main/Main10	Optional
RA	Main/Main10	Main/Main10	Main/Main10	Main/Main10	N/A	Optional
LDB	N/A	Main/Main10	Main/Main10	Main/Main10	Main/Main10	Optional
LDP	N/A	Optional	Optional	Optional	Optional	Optional

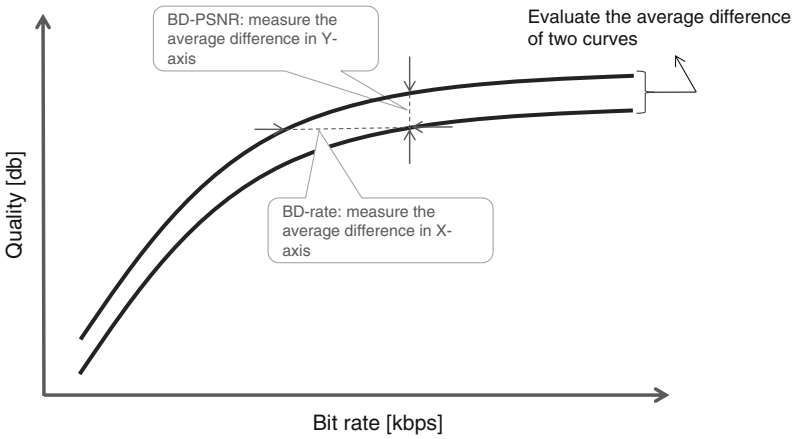


Fig. 9.4 An example of R–D curve

9.2.6 Rate Distortion Curves

When evaluating the coding performance of a video codec, a graph of R–D curve (Rate–Distortion Curve) is used. R–D curve is generated by plotting the encoded results, in terms of bit rate versus the resulting quality, in a graph. The horizontal axis denotes the bit rate and the vertical axis denotes a measure of distortion or quality of encoded video. In general, a higher compression ratio results in a lower bit rate; however, picture quality is generally reduced. Low compression ratio, on the other hand, improves picture quality but at the cost of an increase in bit rate. Since a high coding efficiency codec can achieve higher quality at lower bit rates, the R–D curve moves toward upper left, as shown in Fig. 9.4.

As an objective measurement of picture quality, PSNR (Peak Signal to Noise Ratio) is widely used. PSNR can be calculated by the following equation.

$$PSNR = 10\log_{10}\frac{\left(2^{bitdepth}-1\right)^2 * W * H}{\sum_i \{O_i - D_i\}^2}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00291

where

bitdepth: Bit depth of each pixel
W: Number of horizontal pixels
H: Number of vertical pixels
O_i: Pixel value of the reference picture
D_i: Pixel value of the decoded picture
i: Pixel address

PSNR is calculated for each YCbCr component. In YCbCr domain, human visual system is more sensitive to luminance (Y) than to chrominance (Cb or Cr); accordingly, and in practice, PSNR for luminance (PSNR Y) is a more important metric for objective quality measurements.

In order to compare the coding efficiency of a reference codec vs. the one being evaluated, the average difference of the two R–D curves is calculated. The average bit rate difference (difference in horizontal direction) is referred to as BD (Bjøntegaard's Delta) Rate and the average PSNR difference (difference in vertical direction) is referred to as BD PSNR [1].

In order to calculate BD Rate and BD PSNR, the two R–D curves (corresponding to reference and tested codecs) are approximated by the following cubic polynomial.

$$PSNR = a + b * (bit\ rate) + c * (bit\ rate)^2 + d * (bit\ rate)^3$$

Parameters a–d in the above equation can be derived by using four data points (PSNR and bit rate points). This polynomial approximation will then allow us to derive the BD Rate by integrating the difference of two curves in horizontal direction and BD PSNR by integrating the difference of two curves in vertical direction (see Fig. 9.4).

BD Rate and BD PSNR have been widely used to evaluate coding tools, in the HEVC standardization work. It is however known that such approximation could sometimes lead to large errors, especially for large pictures (e.g. class A sequences). To further improve the approximation accuracy, a piece-wise cubic interpolation is proposed as an alternative [2].

9.2.7 R–D Optimization

HEVC encoder flexibility stems from the fact that it contains an increased number of coding tools, beyond those provided by earlier video coding standards e.g., AVC. This added flexibility allows an encoder to adaptively determine block dependent coding parameters in terms of:

1. Coding unit (CU) quadtree structure, prediction unit (PU) partition modes and transform unit (TU) quadtree structure;
2. Intra PU prediction mode;

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00292

3. Inter PU motion parameters and reference list index or indices, for motion estimation;
4. Rate–distortion optimized quantization (RDOQ), for quantization process.

The key function and differentiation point of a "good" encoder is the selection of the "best" coding parameters (or so-called syntax element values), for improved coding efficiency. Finding the "best" coding parameters is traditionally performed in a rate–distortion (R–D) sense: it enables tradeoffs between the numbers of bits used to encode a block of the picture vs. the resulting distortion that is produced by using that number of bits. An R–D optimization problem can in general be formulated as:

$$\min_{(coding\ parameters)} (D)\ subject\ to \quad R \leq R_T \tag{9.1}$$

where

$$D = Distortion,$$
$$R = Rate\ \ (number\ of\ bits\ required\ to\ signal\ coding\ parameters)$$
$$R_T = Target\ Rate$$

The above minimization is over a combined set of coding parameters and the distortion term is used to quantify the fidelity between original and reconstructed block. In principal, distortion can be measured either by relying on a mathematical distance or by taking into account perception mechanisms. Perceptual metrics correlate well with viewers' perceptual experience but defining them is challenging because of the complexity of modeling various physiological components involved in human visual system. Objective quality measures based on mathematical distances, on the other hand, are easier to derive and under many circumstances they can still provide good tradeoffs between subjective quality and rate used. They are, moreover, "content-agnostic". That is, the same error distribution on different content could yield similar objective quality metrics. Examples of distance based objective quality metrics include mean-squared error (MSE), peak-signal-to-noise (PSNR), and sum of absolute differences (SAD).

Constrained optimization problem in (9.1) can be turned into an unconstrained optimization problem by the introduction of non-negative Lagrangian multiplier λ which combines R and D into a so-called Lagrangian cost function [20, 22], namely:

$$\min_{(coding\ parameters)} J = (D + \lambda * \mathrm{R}) \tag{9.2}$$

Note that λ acts, in a sense, as a "knob": changing the value of λ enables tradeoffs between rate decreases vs. distortion increases. For example, $\lambda = 0$, in (9.2), corresponds to minimizing distortion; conversely, choosing a large value for λ corresponds to rate minimization. A natural question that arises is what value to choose for λ? Sullivan and Wiegand [22] and Ohm et al. [19] address this question by establishing a relationship between λ and quantization step size Q.

$$\lambda = \mathrm{c} * Q^2 \tag{9.3}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00293

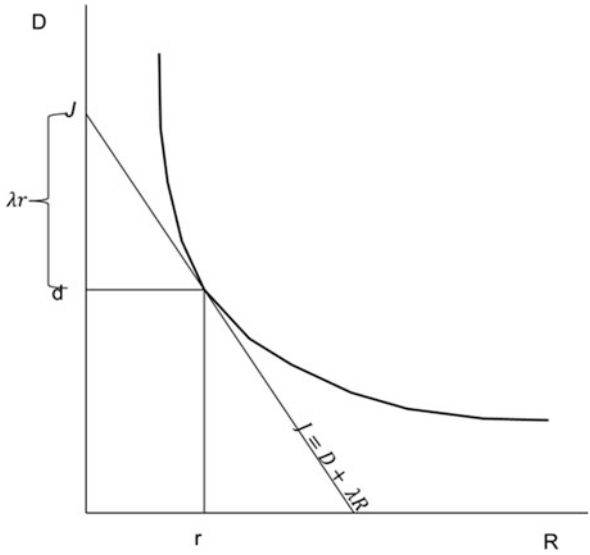


Fig. 9.5 Typical R–D curve and cost function J with slope −λ

In AVC and HEVC, the quantization step size Q is controlled by a quantization parameter (QP) such that Q is proportional to $2^{(QP\text{-}12)/6}$ and the constant of proportionality, c, depends on coding mode decisions.

An example based on a graphical minimization of (9.2) is shown in Fig. 9.5, where a line denoting Lagrangian cost function is plotted against a typical rate–distortion curve that is a non-increasing convex function of R [4]. Minimum J can be achieved by finding the point on the rate–distortion curve which is "hit" first by the plane wave of slope −λ [20].

There are many alternative methods to performing R–D cost optimization. One, for example, can minimize a frame level distortion or minimize an average frame distortion, taken over many video frames. These aforementioned methods are not computationally practical as they will incur significant amount of complexity and delay. Instead, and as described in [15, 19], minimization of (9.2) is performed for each block of samples (e.g., CUs) independently and in four stages: (1) mode decision; (2) intra prediction mode estimation; (3) motion estimation; and (4) quantization. Accordingly, for each block an exhaustive pre-calculation of cost function, associated with each combination of coding parameters, is performed: the optimal R–D solution for the block is the combination that minimizes the R–D cost function. Making block independent assumption despite spatial/temporal dependencies that could exist between blocks (e.g., current block predictor is based on the past reconstructed block samples) is generally ignored for practical applicability [19]. We now describe briefly the four R–D optimization stages:

We let $S_A(i,j)$ and $S_B(i,j)$ denote the $(i,j)^{th}$ sample in blocks A and B, of the same size, respectively. For measuring distortion, we use the following metrics as specified in [15]:

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00294

$$Sum\ of\ Square\ Error\ (SSE) = \sum_{i,j} (s_A(i,j) - s_B(i,j))^2 \tag{9.4}$$

$$Sum\ of\ Absolute\ Difference = \sum_{i,j} |s_A(i,j) - s_B(i,j)| \tag{9.5}$$

$$Hadamard\ Transformed\ SAD\ (SATD) = \sum_{i,j} |HT(i,j)| \tag{9.6}$$

$HT(i,j)$ in (9.6) is the $(i,j)^{th}$ coefficient of a block that is obtained by applying Hadamard transform to the block difference between blocks A and B.

JCT-VC [15] specifies also the following λ values:

$$\lambda_{mode} = \alpha * W_k * 2^{((QP-12)/3.0)} \tag{9.7}$$

$$\lambda_{pred} = \sqrt{\lambda_{mode}} \tag{9.8}$$

$$\omega_{chroma} = 2^{((QP-QP_{chroma})/3.0)} \tag{9.9}$$

$\alpha = 1.0 - Clip3(0.0, 0.5, 0.05 * number\_of\_B\_frames)$ for referenced pictures

$$\alpha = 1.0 \quad \text{for non-referenced pictures} \tag{9.10}$$

where

$$Clip3(x,y,z) = \begin{cases} x; z < x \\ y; z > y \\ z; otherwise \end{cases}$$

Interested readers are referred to [15] for derivation of W_k as well as λ values for chroma.

CU level mode decision (intra vs. inter) coding is based on finding coding parameters that minimize cost function J_{mode} in (9.11).

$$J_{mode} = (SSE_{luma} + \omega_{chroma} * SSE_{chroma}) + \lambda_{mode} * R_{mode} \tag{9.11}$$

Distortion terms SSE_{luma} and SSE_{chroma} correspond to the SSE between the original and reconstructed luma and chroma CU blocks respectively. Similarly, R_{mode} represents the total number of bits used for CU level intra or inter mode signaling, PU partition(s) within the CU, PU prediction mode(s) in case of intra mode or PU motion parameters in case of inter mode, TU quadtree partition(s), and finally number of bits required for representing quantized residual transform coefficient levels.

For finding the best inter CU coding cost, J_{mode} is evaluated for all possible PU partition modes (e.g., 2N × 2N, N × N, 2N × N, N × 2N, nl × 2N, nR × 2N) and a partition that gives the minimum coding cost is chosen.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00295

Motion estimation for each inter PU partition is done based on the minimization of inter prediction cost shown in (4.12).

$$mp^* = arg \min_{mp \in MP} D_{mp} + \lambda_{pred} * R_{mp} \tag{9.12}$$

For a given reference picture list, set MP, over which the minimization is carried out, consists of all possible motion parameters, namely motion vectors and associated reference indices. The minimization task in (9.12) is broken into two parts: integer-sample precision and sub-sample precision. For integer-sample precision, distortion term D_{mp} corresponds to the SAD between original PU block and its motion compensated reference block. For sub-sample motion search however distortion term D_{mp} represents SATD of the block difference between the original and sub-sample motion compensated reference block. R_{mp} term represents an estimate of the number of coded bits required to transmit mp.

For bi-prediction, cost function minimization in (9.12) becomes a joint optimization problem and is solved by the application of an iterative algorithm [5]. The algorithm is initialized first with the two best motion parameters that are obtained independently, for each reference list (L0 and L1). Iteration for further refinement and combined cost minimization is performed by keeping motion parameter of L0 list constant while performing sub-pixel motion search on the complementary list (L1). Once minimum cost is achieved, motion parameter associated with L1 list is held constant and motion parameter of the L0 list is adjusted for computing minimum combined cost. This "ping-pong" like iteration process is continued until convergence is reached.

For intra PU prediction, a two-stage minimization process is performed:

At the first stage, a fix number[1] of candidate intra prediction modes with lowest prediction cost are chosen according to the minimization of the prediction cost function in (9.13).

$$p^* = \arg \min_{p \in P} D_p + \lambda_{pred} * R_p \tag{9.13}$$

Distortion term D_p in (9.13) represents the SATD between the original block and its prediction block using intra prediction mode p and R_p represents number of coded bits required for signaling mode p. Set P, over which minimization is carried out, consists of planar, DC and all the 33 angular prediction directions.

In the second stage, the list containing the candidate intra prediction modes from the first stage is augmented with the three most probable modes if not already present in the list. The best intra prediction mode is the one that gives the minimum J_{mode} among candidate intra prediction modes in this augmented list.

Note that HEVC allows PCM coding of a CU block if the block size is greater or equal to a signaled minimum PCM coding block size. For PCM J_{mode} evaluation,

[1]These fix values are pre-determined and they depend on PU size.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00296

distortion terms SSE_{luma} and SSE_{chroma} are set to zero when both input and PCM coded samples have the same bit depth. Term R_{mode} includes all the bits required for signaling PCM mode and PCM coded samples.

Finally, by applying this CU level mode decision at each level of CU recursion tree a coding tree unit (CTU) level coding mode decision can be obtained.

The goal of Rate distortion optimized quantization (RDOQ), in quantization process stage, is the adjustment of transform coefficient levels, in R–D sense [17]. For an insight to the general concept of minimization process, assume c_k to be the last non-zero coefficient in a transformed block for a given position k; then, for each transform coefficient level l_i, at position $\mathrm{i} = \mathrm{k} - 1, \ \ldots \ 0$, RDOQ tries to find the optimal transform coefficient level, l_i^* that minimizes the cost function, $J_k(l_i)$, below:

$$J_k\left(l_i\right) = D_k\left(l_i\right) + \lambda * \mathrm{R}_k\left(l_i\right) \tag{9.14}$$

For computational simplicity, possible values of l_i are limited to be either zero, or truncated l_i, or rounded-up l_i (i.e., l_{floor} and $l_{ceiling}$). Distortion term $D_k\left(l_i\right)$ is due to the quantization error and is calculated as normalized SSE in transform domain and $R_k\left(l_i\right)$ denotes number of bits used for transmitting level l_i. The optimal solution is the vector of re-quantized transform levels at position k^* with minimum J_k over all possible positions, k.

9.3 Objective Performance Analysis

This section summarizes the comparison of coding efficiency of HEVC and AVC. The test conditions are summarized in Table 9.4 and all encoders settings described in Table 9.3 are used for the comparisons.

The results for the test sequences in Table 9.1 are summarized in Tables 9.5, 9.6, 9.7 and 9.8. In case of Random Access Main, coding efficiency of HEVC is 42.7 % higher than that of AVC. In case of All Intra Main however the improvement is 21.9 % which indicates that the improvement in Intra picture is lower than that in predictive pictures (P or B picture).

As an example, R–D curves of the sequence, Four People (Class E, RA Main) are shown in Figs. 9.6, 9.7, and 9.8.

Table 9.4 Test conditions

Test conditions	
Encoder	HM12.1 (HEVC) and JM18.5 (AVC)
Test sequences	All sequences defined in Table 9.1
Bit depth	8-bit (Main configuration)
Prediction structure	AI, RA, LDP and LDB
QPI	22, 27, 32 and 37

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00297

Table 9.5 Comparison of coding performance of HEVC and AVC (All Intra Main)

	All Intra Main		
	Y (%)	U (%)	V (%)
Class A	−23.6	−21.1	−19.9
Class B	−22.7	−22.1	−21.7
Class C	−19.7	−20.8	−21.1
Class D	−16.4	−17.0	−17.7
Class E	−28.8	−27.1	−27.1
Overall	**−21.9**	**−21.4**	**−21.2**
Class F	−28.6	−25.3	−26.2

Table 9.6 Comparison of coding performance of HEVC and AVC (Random Access Main)

	Random Access Main		
	Y (%)	U (%)	V (%)
Class A	−42.6	−42.5	−44.5
Class B	−47.7	−43.3	−42.5
Class C	−35.5	−34.7	−35.4
Class D	−33.9	−36.6	−37.9
Class E	−56.0	−54.4	−55.8
Overall	**−42.7**	**−41.7**	**−42.5**
Class F	−53.1	−52.2	−53.9

Table 9.7 Comparison of coding performance of HEVC and AVC (Low Delay B Main)

	Low Delay B Main		
	Y (%)	U (%)	V (%)
Class A	−38.6	−24.9	−26.7
Class B	−42.1	−34.4	−35.3
Class C	−32.7	−32.2	−32.9
Class D	−29.9	−31.2	−33.0
Class E	−44.1	−39.2	−38.8
Overall	**−36.6**	**−32.6**	**−33.6**
Class F	−33.9	−35.3	−37.8

Table 9.8 Comparison of coding performance of HEVC and AVC (Low Delay P Main)

	Low Delay P Main		
	Y (%)	U (%)	V (%)
Class A	−29.3	−31.4	−33.4
Class B	−38.1	−37.3	−39.6
Class C	−32.4	−38.2	−38.5
Class D	−30.1	−38.2	−38.6
Class E	−44.4	−44.9	−44.7
Overall	**−35.3**	**−38.0**	**−38.9**
Class F	−35.3	−39.5	−40.5

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00298

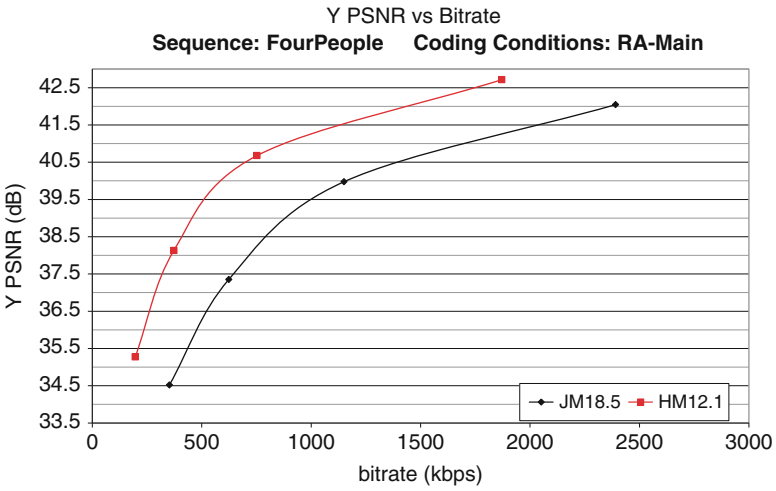


Fig. 9.6 R–D curve of Y (Four People, RA-Main)

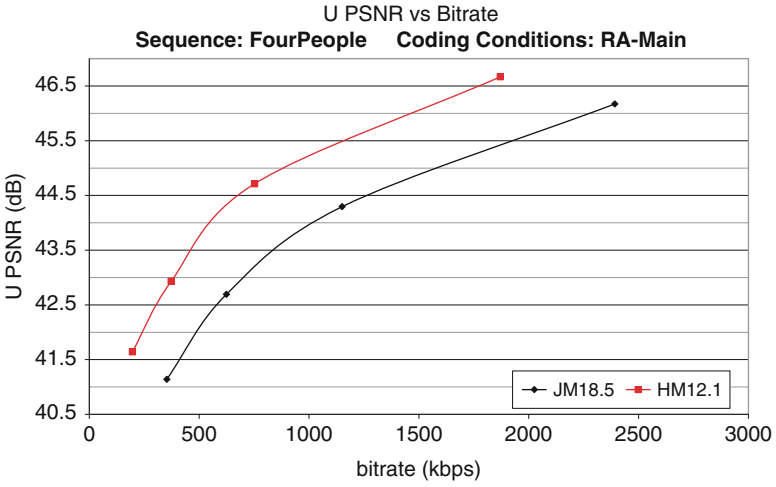


Fig. 9.7 R–D curve of U (Four People, RA-Main)

The results for 4K test sequences in Table 9.2 are summarized in Table 9.9 (Random Access Main only). We observe a coding efficiency improvement of up to 76 % for HEVC.

In addition, still picture coding performance of HEVC based intra coding, relative to JPEG and AVC intra coding is reported in [18]. The results show that bit rate reductions due to HEVC intra coding are about 44 and 32 %, respectively. Comparisons of HEVC intra coding to JPEG and JPEG2000 by means of objective and subjective evaluations are also reported in [9]. The evaluation

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00299

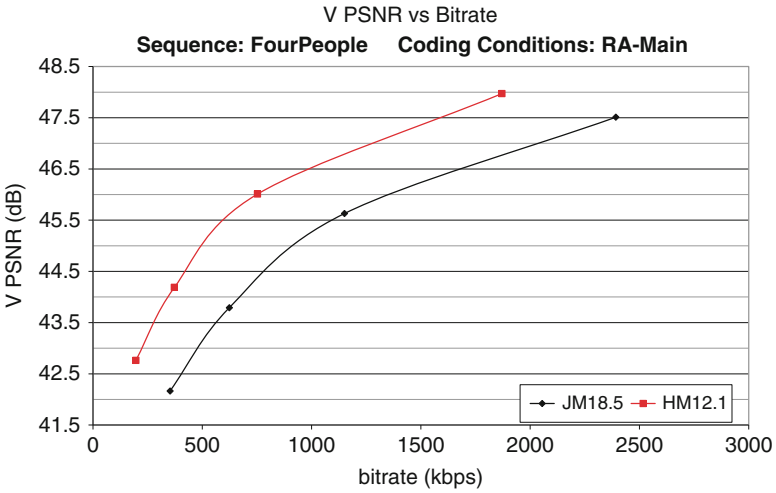


Fig. 9.8 R–D curve of V (Four People, RA-Main)

Table 9.9 Comparison of coding performance of HEVC and AVC (Random Access Main)

	Random Access Main		
	Y (%)	U (%)	V (%)
Book	−58.8	−55.2	−56.7
BT709Birthday	−60.4	−54.3	−57.6
HomelessSleeping	−75.9	−79.6	−83.5
Menage	−33.3	−34.1	−36.2
Trafftc	−41.4	−43.7	−43.6

results demonstrate that HEVC intra coding outperforms encoders for still images with an average bit rate reduction ranging from 16 % (compared to JPEG 2000 4:4:4) up to 43 % (compared to JPEG).

9.4 Subjective Performance Analysis

Because, subjective evaluation of video content correlates directly with the viewer perceptual experience, it could very well be considered as a more reliable performance measure of a codec. It is therefore important that for conducting subjective evaluation test, the testing methodology be defined in accordance with the universally accepted guidelines and practices, such as those described in Recommendation ITU-R BT.500 [11–14]. In the following sub-sections, we will further elaborate on the testing methodology and environment together with references to subjective evaluation test results.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00300

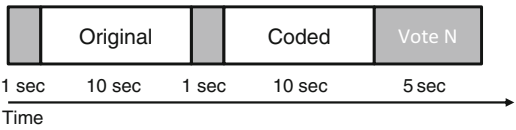


Fig. 9.9 DSIS basic test cell (BTC)

9.4.1 Test Methodology

First, we provide a brief tutorial about some frequently used subjective quality assessment methods. In general, there are two broad methods to carry visual evaluation tests: double stimulus and single stimulus. In double stimulus test subjects rate either the quality or change in the quality between two video clips reference (original) vs. impaired (coded). In single stimulus test, subjects rate the quality of the impaired (coded) video clip, only. We will now describe two examples of the former, namely, double stimulus impairment scale (DSIS), and double stimulus continuous quality scale (DSCQS).

9.4.1.1 DSIS (Double Stimulus Impairment Scale)

This method is used when the material to be evaluated shows a wider range of visual quality covering all quality scales (and not of the impairments). There are two variants of DSIS: Variant I and Variant II. The structure of the Basic Test Cell (BTC) of Variant I, is shown in Fig. 9.9. It consists of two consecutive presentations of video clips. Original (reference) video clip is presented first followed by the presentation of the impaired (coded) version of the video clip. A message is then displayed for 5 s requesting viewers to vote.

Viewers are expected to mark their visual quality score on an answer sheet with quality rating over a defined scale e.g., scale that is made of 5 levels—ranging from "1" (very annoying) to "5" (imperceptible). In Variant II of DSIS, the pairs of original (reference) video clip and impaired (coded) version of the video clip are presented twice before voting. For visual test evaluations conducted in Sect. 9.4.2, Variant I of DSIS methodology, as described earlier was chosen.

9.4.1.2 DSCQS (Double Stimulus Continuous Quality Scale)

Double Stimulus Continuous Quality Scale (DSCQS) is used in cases when it is not possible to present the full range of quality scales. In this method, the original (reference) and the coded (impaired) samples of a video clip are presented twice and, in random order, for each BTC. At the end of the second presentation, the viewers are asked to grade each of the two original and the two coded video clips, separately. It should be noted that because of the random presentation order, viewers do not have an *a priori* knowledge of whether a video clip shown belongs to the original or to the impaired one.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00301

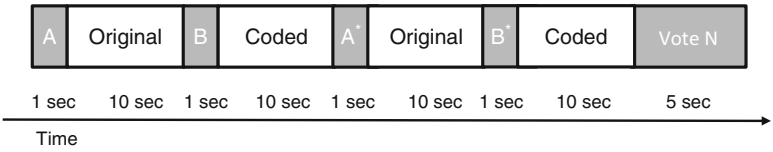


Fig. 9.10 DSCQS basic test cell (BTC)

As shown in Fig. 9.10, the BTC structure of the DSCQS method contains two consecutive pairs of presentations. At first, a mid-grey screen with the letter "A", in the middle, is displayed for a second followed by a 10-s presentation of a video clip—either original or impaired. Then, a mid-grey screen with the letter "B" appears followed by a 10-s presentation of the second video clip. Similar process is repeated during the second round of presentation by changing letters A and B to A* and B*, instead. Finally, a message is displayed for 5 s instructing the viewers to vote.

9.4.1.3 Training Session

The outcome of the visual tests could be highly dependent on the proper training of the participants. In order to allow viewers to get familiarized with the testing procedures, it is important that viewers are briefed about the testing procedures and participate in a training session before starting subjective evaluation tests. Also, the video clips shown for the training need to be different from those used during the actual tests. Coding impairments should resemble those that appear on the tested materials, though. In the training session, three BTCs (the worst quality, medium quality and the best quality) should be included allowing viewers to know the quality range of the test.

9.4.1.4 Viewing Environment

In the laboratory where the viewing session is being held, general internal light has to be low and a uniform light has to be placed behind the monitor. The intensity of the light is specified in the ITU-R BT.500 [11, 14]. No light source has to be directed to the screen or cause reflections. Ceiling, floor and walls of the laboratory have to be made of non-reflecting material (e.g. carpet or velvet) and should have a color tuned as close as possible to CIE Standard Illuminant D65 (daylight illuminant, 6500K). The viewing room must be protected from external visual or audio pollution.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00302

9.4.2 Subjective Quality Evaluation Test

This section reports the results of subjective quality evaluation conducted at EPFL's MMSPG test laboratory, which fulfills the recommendations for the subjective evaluation of visual data issued by ITU-R BT.500 [11, 14]. It is also worth noting that the testing methodology performed in this section has benefited significantly from the experience gained while conducting the subjective evaluation tests described in [8].

9.4.2.1 Test Environment

The test room is equipped with a controlled lighting system with a 6,500 K color temperature and an ambient luminance at 15 % of the maximum screen luminance, whereas the color of all the background walls and curtains present in the test area are in mid grey. The laboratory setup is intended to ensure the reproducibility of the subjective tests results by avoiding unintended influence of external factors.

To display the test stimuli, two Eizo CG301W LCD monitors with a native resolution of $2{,}560 \times 1{,}600$ pixels were used. The monitors were calibrated using an X-Rite i1Display Pro color calibration device according to the following profile: sRGB gamut, D65 white point, 120 cd/m^2 brightness, and minimum black level.

The experiment involved two subjects per monitor assessing the test material. The subjects were seated in a row perpendicular to the center of the monitor, at a distance of 2.2 times the picture height, roughly corresponding to a visual angle of 1 arc-minute between two adjacent pixels, as suggested in [13].

9.4.2.2 Test Methodology

The double stimulus impairment scale (DSIS Variant I) methodology as described earlier was chosen for the testing. A five-grade impairment scale (5: Imperceptible, 4: Perceptible but not annoying, 3: Slightly annoying, 2: Annoying, 1: Very annoying) was used. The subjects were presented with pairs of video sequences (i.e., stimuli), where the first sequence was always a reference video (stimulus A) and the second, the video to be evaluated (stimulus B). After the presentation of each pair of sequences, a 5-s voting time followed. Subjects were asked to rate the impairments of the second stimulus in relation to the first stimulus, and to express these judgments in terms of the wordings used to define the rating scale.

9.4.2.3 Dataset

Five video sequences in Table 9.2 were used in the experiments, with different visual characteristics, resolutions, and frame rates. All sequences were stored as

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00303

raw video files, progressively scanned, and with YCbCr 4:2:0 color sampling. The sequences were compressed with HEVC and AVC. For each sequence and codec, four quantization parameters were selected, resulting in a total of 40 test stimuli.

Five training samples were generated using the Sintel39 sequence (its resolution is $3{,}840 \times 1{,}744$) and manually selected by expert viewers so that the quality of samples were representative of all grades of the rating scale.

The original sequences were cropped to the resolution of the monitor, keeping only the central part, and the 10-bit sequences were clipped to 8-bit.

9.4.2.4 Training Session

Before the experiment, a consent form was handed to subjects for signature, and oral instructions were provided to explain their tasks. Additionally, a training session was organized to allow subjects to familiarize with the assessment procedure.

9.4.2.5 Test Session

Since the total number of test samples was too large for a single test session, the overall experiment was split into two sessions of approximately 13 min each. Between the sessions, the subjects took a 10 min break. The test material was randomly distributed over the two test sessions.

Three dummy pairs (one with high quality, one with low quality, and one of mid quality), whose scores were not included in the results, were included at the beginning of each test session to stabilize the subjects' ratings. To reduce contextual effects, the stimuli orders of display were randomized applying different permutation for each group of subjects, whereas the same content was never shown consecutively.

A total of 18 naive subjects (6 females and 12 males) took part in the experiments. They were between 18 and 27 years old with an average of 23.4 years of age. All subjects were screened for correct visual acuity and color vision using Snellen and Ishihara charts, respectively.

9.4.2.6 Analysis of the Results

The subjective results were processed by first detecting and removing subjects whose scores appeared to deviate strongly from others. The outlier detection was performed according to the guidelines described in Section 2.3.1 of Annex 2 of [14]. In this study, one outlier was detected. Then, the mean opinion score (MOS) was computed for each test stimulus as the mean across the rates of the valid subjects, as well as associated 95 % confidence interval (CI), assuming a Student's *t*-distribution of the scores.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00304

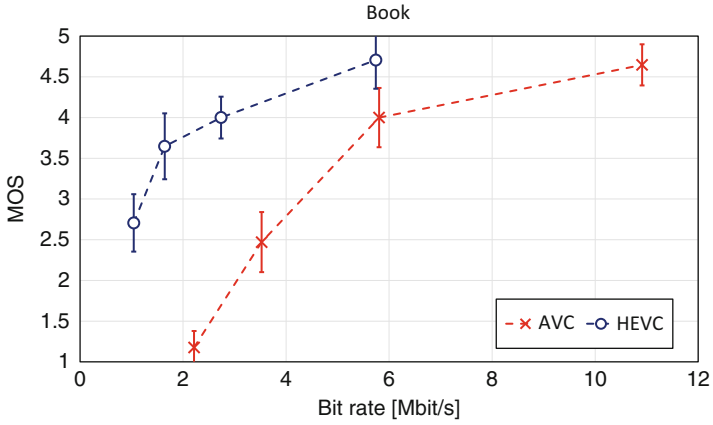


Fig. 9.11 R–D curve (Book)

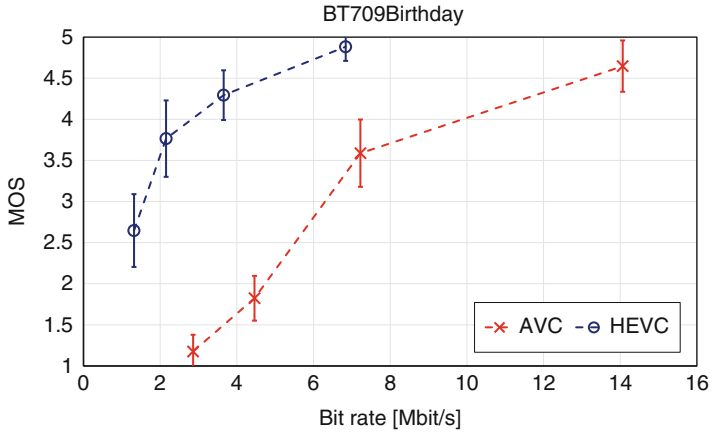


Fig. 9.12 R–D curve (BT709Birthday)

9.4.2.7 Rate Distortion Curves Results

The R–D curves obtained by the subjective quality evaluation are shown in Figs. 9.11, 9.12, 9.13, 9.14 and 9.15.

From these figures, it can be seen that HEVC shows substantial visual quality improvements over AVC, especially at lower bit rates.

9.4.2.8 Average Bit Rate Difference

The average bit rate difference for HEVC over AVC was computed using the model proposed in [7]. This model is an extension of the Bjøntegaard model [1]

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00305

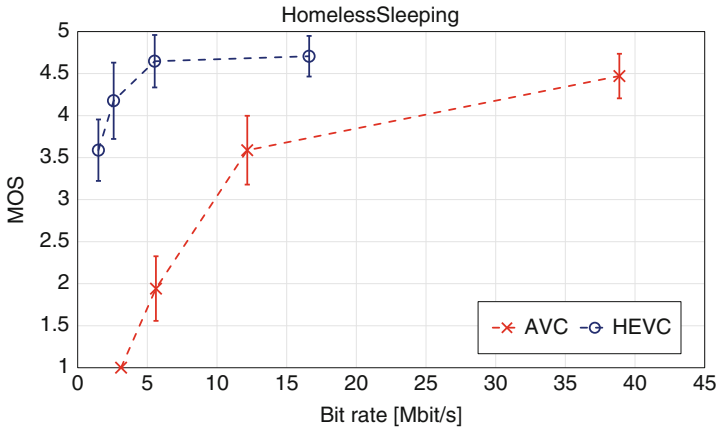


Fig. 9.13 R–D curve (HomelessSleeping)

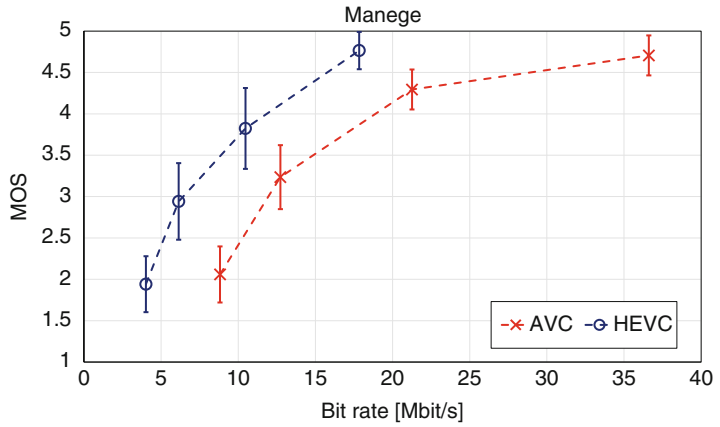


Fig. 9.14 R–D curve (Manege)

for subjective scores: ΔR is computed from the MOS; $[\Delta R_{min}, \Delta R_{max}]$ provide a confidence interval on ΔR and is determined considering the confidence index (CI) computed on the subjective scores; the confidence index takes into account the spreading of the MOS over the rating scale and the goodness of the fit of the values (Table 9.10).

For visual quality evaluation of CTC test sequences, interested readers are referred to [19], in which results of subjective tests are reported. The reported results indicate that a bit rate reduction of 50 % can be achieved for the example video test set.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00306

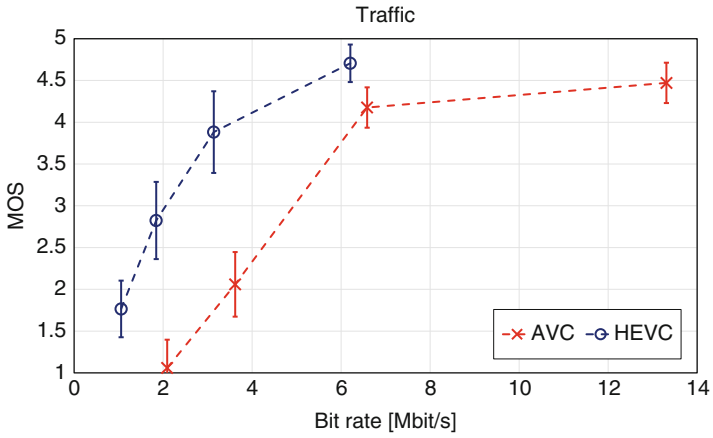


Fig. 9.15 R–D curve (Traffic)

Table 9.10 Bit rate differences of tested bitstreams

Sequence	ΔR (%)	[ΔR_{min}, ΔR_{max}]	Confidence index (%)
Book	−62	[−75 %, −51 %]	100
BT709Birthday	−71	[−81 %, −60 %]	100
HomelessSleeping	−87	[−94 %, −71 %]	100
Manege	−43	[−60 %, −17 %]	88
Traffic	−55	[−68 %, −39 %]	100
Average	−64	[−76 %, −48 %]	98

9.5 Production–Quality Encoder Performance Analysis

This section presents the results of an informal subjective quality comparison between the eBrisk-UHD and x264 [23] production–quality encoders, which were configured to be conformant with HEVC Main Profile and AVC High Profile, respectively. The encoder comparison presented in this section is intended to complement the subjective quality comparisons discussed earlier in this chapter in which HEVC and AVC encoder reference software were used.

9.5.1 Test Conditions

In this section, the test conditions, including the encoder configuration and evaluation conditions (e.g., sequence presentation details, viewing equipment, lighting conditions), and tested video sequences are described.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00307

Table 9.11 Video sequences used for the subjective quality comparison in order of presentation (top to bottom)

Resolution	Sequence name	Frame count	Frame rate (fps)	Bit depth
3,840 × 2,160	IntoTrees	400	50	8
3,840 × 2,160	OldTownCross	400	50	8
3,840 × 2,160	ParkJoy	400	50	8
3,840 × 2,160	DucksTakeOff	400	50	8

9.5.1.1 Encoder Settings

The encoders were configured for high coding efficiency operation. More specifically, the HEVC encoder was configured for Main profile and to use a prediction structure similar to that described in Sect. 9.2.3.2 with the period of the intra pictures set to 48 frames. The AVC encoder was configured to use default parameter values except for those parameters that required an explicit setting (e.g., the keyint parameter, used to specify the intra frame period, was set to 48). Several AVC encodings were performed for each video sequence, each with a different quantization parameter (QP) value. In each case, the encoding that yielded an average bit rate closest to 2.5 times that of the corresponding HEVC encoding was selected (i.e., the bit rate of the HEVC encoded sequence was approximately 60 % lower than that of the AVC encoded sequence).

9.5.1.2 Subjective Evaluation Conditions

Twenty-seven (27) volunteer viewing subjects were used for the subjective experiments. Fifteen (15) of the viewers had little or no previous experience in evaluating video sequences. The viewing was conducted in a somewhat-darkened room using a XBR55X900A 55″ UHD Sony Bravia LED monitor. Additional key elements of the subjective evaluation and video presentation methodology are listed below:

1. Untrained viewers participated one at a time with each viewer seated in a chair that was positioned approximately 1.5 meters from the monitor and centered.
2. For each of the four test sequences listed in Table 9.11, the HEVC and AVC encoded video sequences were cropped in the horizontal dimension and spliced together side-by-side.[2] The relative position of the compared encodings was randomized and the experiments were executed in a double-blind manner.
3. The sequences were displayed at their native spatial resolutions; however, the display rate was set to 30 frames per second (fps) (i.e., the sequences were displayed at 60 % of their native 50 fps frame rate).

[2]The horizontal cropping removed one half of the pixels so as to enable both the HEVC and AVC encoded sequences to be displayed side-by-side on the 3840 pixel-wide 4K monitor.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00308

4. The video sequences were presented to each viewer between one and three times and the viewers were asked to assess the relative quality of the side-by-side encodings according to a 5° scale: *left better, left slightly better, no preference, right slightly better* or *right better*.[3] The viewers were provided approximately 30 seconds between the presentation of each of the spliced video sequences to record their preferences. A total of seven (7) video pairs were presented and each viewing session had a duration of approximately seven (7) minutes.

9.5.1.3 Test Sequences

The four video sequences described in Table 9.11 were used in the subjective experiments. The YCbCr 4:2:0 8-bit 4K video sequences have a variety of characteristics typical of what might be encountered in the context of a streaming application, and they were selected from those that are generally available and used for video coding test purposes. All the sequences have a native frame rate of 50 fps. The 4K sequences were displayed at 30 frames per second; the highest frame rate at which the monitor is capable of displaying 4K content. The order of presentation followed that shown in Table 9.11, from top to bottom.

9.5.2 Subjective Quality Assessment Results

Table 9.12 shows the results of subjective video quality assessments. For each of the video sequences, the average bit rates of the HEVC and AVC encoders are shown along with the viewers' assessments of the video sequences, according to the 5° scale described in Sect. 9.5.1.2. Each video sequence was encoded using a fixed QP. The QP values for the encodings were selected to yield good, but not especially high video quality, in order to avoid viewing scenarios where either both encodings would yield indistinguishably excellent quality or both encodings would yield substantial coding artifacts.

9.5.3 Results

The subjective results presented in Table 9.12 show that the viewers either had no preference or favored the HEVC encoded video at a bit rate that was approximately

[3] A 5° scale was selected for this study in lieu of more commonly used 7° and 3° scales, because it is known to work well to prioritize video quality improvement work during commercial encoder development.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00309

Table 9.12 Subjective viewing comparison results for sequences encoded using the HEVC and AVC encoders (*B* better, *SB* slightly better, *NP* no preference)

Sequence name	HEVC QP	HEVC bit rate	AVC QP	AVC bit rate	HEVC B	HEVC SB	NP	AVC SB	AVC B
InToTrees	34	4.5 Mbps	35	11.9 Mbps	21	6	0	0	0
OldTownCross	32	3.1 Mbps	34	7.8 Mbps	14	3	3	4	3
ParkJoy	38	11.9 Mbps	38	29.0 Mbps	0	2	7	10	8
DucksTakeOff	37	15.4 Mbps	39	37.6 Mbps	12	4	3	6	2
Selection count totals					47	15	13	20	13
Percent totals (%)					43.5	13.9	12.0	18.5	12.0

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00310

60 % lower than that of AVC in 69.4 % of the trials.[4] These results are consistent with the subjective results reported in [8]. In addition, comparing the coding efficiency gains of HEVC relative to AVC for certain 4K sequences in this study with those gains reported in earlier studies (e.g., [10]), in which high-quality resampled (lower-resolution) versions of the same video sequences were used, it can be seen that the coding efficiency gains of HEVC relative to AVC are larger for the 4K sequences.[5] This comparison suggests that the increased coding efficiency gains for HEVC compared with AVC observed for 4K sequences cannot be explained solely by differences in content.

9.6 Conclusions

In this chapter, performance analysis of HEVC in comparison with AVC in terms of objective as well as subjective quality assessments are given. Because of the increased flexibility offered by HEVC, methods to select the best coding parameters, in a rate–distortion sense, are also described. Special care has been taken to apply a unified approach when conducting subjective and objective quality evaluations between HEVC and AVC. Both objective and subjective tests results indicate significant gains in compression efficiency of HEVC over AVC. More specifically, the bit rate reduction, based on objective evaluation of CTC test sequences, indicates an overall performance improvement of about 22 % for AI, 43 % for RA, 37 % for LDB and 35 % for LDP over AVC. Furthermore, by using non-CTC test sequences, we observe up to 76 % improvement in coding efficiency, as indicated in Table 9.9. Results of subjective evaluation tests indicate that an even higher bit rate saving in the ranges of 55–87 % can be achieved. The informal visual quality evaluation test results also confirm that HEVC yields a substantial improvement in compression capability beyond that of AVC for video streaming applications. It is also suggested that the coding performance gains of HEVC over AVC generally increase with increasing video resolution up to at least 4K resolutions.

Acknowledgments Regarding Sect. 9.4, the authors wish to thank 4EVER consortium, British Broadcasting Corporation (BBC), Kamerawerk GmbH and Technicolor for providing the original video test sequences used for visual testing. Our special thanks and gratitude go also to T. K. Tan, of NTT Docomo, for his help and support in the preparation of coded video test sequences used for subjective performance evaluations in Sect. 9.4. Subjective evaluations performed at EPFL were possible thanks to the Swiss National Foundation for Scientific Research (FN 200021-143696-1), EC funded Network of Excellence VideoSense, and COST Action IC1003 European Network on Quality of Experience in Multimedia Systems and Services QUALINET. Efforts by Hiromi Nemoto for set up and conducting subjective evaluations at EPFL are also acknowledged.

[4]The 69.4 % result was derived by summing the selection counts in the HEVC B, HEVC SB and NP columns of Table 9.12 and dividing by the total number of selection counts.

[5]The earlier studies also used earlier versions of the HEVC draft standard and a slightly different evaluation methodology.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00311

References

1. Bjøntegaard G (2001) Calculation of average PSNR differences between RD-curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001
2. Bossen F (2011) Excel template for BD-rate calculation based on Piece-wise Cubic Interpolation, JCT-VC Reflector
3. Bossen F (2013) Common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L1110, Geneva, Jan. 2014
4. Cover TM, Thomas JA (1991) Elements of information theory, Chapter 13. Wiley, New York
5. Flierl M, Girod B (2003) Generalized B pictures and the draft H.264/AVC video compression standard. IEEE Trans Circuits Syst Video Technol 13(7):587–597
6. H.264/MPEG-4 AVC Reference Software, Joint Model 18.5: http://iphome.hhi.de/suehring/tml/download/jm18.5.zip
7. Hanhart P, Ebrahimi T (2014) Calculation of average coding efficiency based on subjective quality scores. J Visual Commun Image Represent 25(3):555–564
8. Hanhart P, Rerabek M, De Simone F, Ebrahimi T (2012) Subjective quality evaluation of the upcoming HEVC video compression standard. In: *Proc. SPIE.* 8499, Applications of Digital Image Processing XXXV, no. 84990V, Oct. 2012
9. Hanhart P, Rerabek M, Korshunov P, Ebrahimi T (2013) Subjective evaluation of HEVC intra coding for still image compression. In: Seventh international workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM), Scottsdale, Arizona
10. Horowitz M, Kossentini F, Mahdi N, Xu S, Guermazi H, Tmar H, Li B, Sullivan GJ, Xu J (2012) Informal subjective quality comparison of video compression performance of the HEVC and H.264/MPEG-4 AVC standards for low-delay applications. In: *Proc. SPIE.* 8499, Applications of Digital Image Processing XXXV, no. 84990W, Oct. 2012
11. ITU-R Rec. BT.500-11 (2006) Methodology for the subjective assessment of the quality of television pictures
12. ITU-T Rec. P.910 (2008) Subjective video quality assessment methods for multimedia applications
13. ITU-R BT.2022 (2012) General viewing conditions for subjective assessment of quality of SDTV and HDTV television pictures on flat panel displays
14. ITU-R BT.500-13 (2012) Methodology for the subjective assessment of the quality of television pictures
15. McCann K, Bross B, Han WJ, Kim IK, Sugimoto K, Sullivan GJ (2013), High Efficiency Video Coding (HEVC) Test Model 13 (HM 13) Encoder Description, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-O1002, Geneva, Oct. 2013
16. Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11. HM12.1 Reference Software: https://hevc.hhi.fraunhofer.de/svn/svn-HEVCSoftware/(mainsite) and http://hevc.kw.bbc.co.uk/svn/jctvc-a124/(mirrorsite)
17. Karczewicz M, Ye Y, Chong I (2008) Rate distortion optimized quantization, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-AH21, Antalya, Jan. 2008
18. Nguyen T, Marpe D (2012) Performance analysis of HEVC based Intra coding for still image compression. In: PCS2012, May 2012, pp 233–236
19. Ohm J-R, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards –including High Efficiency Video Coding (HEVC). IEEE Trans Circuits Syst Video Technol 22(12):1669–1684
20. Ortega A, Ramchandran K (1999) Rate-distortion methods for image and video compression: an overview. IEEE Signal Process J 23–50
21. Schwarz H, Marpe D, Wiegand T (2005) Hierarchical B pictures, Joint Video Team (JVT), Document JVT-P014, Poznan, July 2005
22. Sullivan GJ, Wiegand T (1999) Rate-distortion optimization for video compression. IEEE Signal Process J 74 -90
23. VideoLANx264 SW library (2013) http://www.videolan.org/developers/x264.html. Version core l35 r2345, 30 July 2013

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00312

Chapter 10
Decoder Hardware Architecture for HEVC

Mehul Tikekar, Chao-Tsung Huang, Chiraag Juvekar, Vivienne Sze, and Anantha Chandrakasan

Abstract This chapter provides an overview of the design challenges faced in the implementation of hardware HEVC decoders. These challenges can be attributed to the larger and diverse coding block sizes and transform sizes, the larger interpolation filter for motion compensation, the increased number of steps in intra prediction and the introduction of a new in-loop filter. Several solutions to address these implementation challenges are discussed. As a reference, results for an HEVC decoder test chip are also presented.

10.1 Introduction

HEVC presents several new challenges for a hardware decoder implementation. HEVC's decoding complexity is found to be between 1.4× and 2× of H.264/AVC [22] when measured in terms of cycle count for software. In hardware, however, the increased complexity of HEVC entails significant increase in hardware cost over traditional H.264/AVC decoders, both at the top-level of the video decoder, and in the low-level processing blocks. Some of the challenges are listed below.

- The diverse sizes of Coding Tree Units (CTU), Coding Units (CU), Prediction Units (PU) and Transform Units (TU) require complex state machines to control the system pipeline and data paths in the individual processing blocks.
- The largest CTU (64 × 64) is 16× larger than the H.264/AVC macroblock (16 × 16), which means that the memories in pipeline stages need to be proportionately larger.

M. Tikekar (✉) • C. Juvekar • V. Sze • A. Chandrakasan
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
e-mail: mtikekar@mit.edu

C.-T. Huang
National Tsing Hua University, Hsinchu, Taiwan

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_10
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00313

- The inverse transform block is considerably more complicated due to the large TU sizes and higher precision of the transform matrix. The largest TU size (32 × 32) requires a 16× larger transpose memory.
- HEVC uses an 8-tap luma interpolation filter for motion compensation as compared to the 6-tap filter in H.264/AVC. This increases the bandwidth required from the decoded picture buffer.

The architecture of the video decoder depends strongly on parameters such as the required throughput (i.e. pixel rate defined by the level limit in the HEVC specification), technology node, area and power budgets, control and data interface to the external world and memory technology used for the decoded picture buffer. In this chapter, we describe the architecture for an HEVC decoder for 4K Ultra HD decoding at 30 fps designed in 40 nm CMOS technology with external DDR3 memory for the decoded picture buffer. The decoder operates at 200 MHz and is frequency-scalable for lower resolutions and picture rates. Along with techniques used in H.264/AVC decoders, such as frame-level parallelism [29] and reference frame compression [20], and general VLSI techniques such as pipelining and dynamic voltage and frequency scaling, HEVC decoders can benefit from architectural techniques like:

- Variable-size pipelining to reduce on-chip SRAM and handle different CTU sizes.
- Unified processing engines for prediction and transform to manage the large diversity of PU and TU sizes.
- High-throughput motion compensation (MC) cache to address increased DRAM requirements for the longer interpolation filters.

10.2 System Pipeline

The granularity of the top-level pipeline is affected by processing dependencies between pixels. For example, computing the luma residue at any pixel location requires all transform coefficients in the TU that contains the pixel. Hence, it is not possible for the inverse transform block to use, say, a 4 × 4 pixel pipeline; the pipeline granularity must be at least one TU in size. In general, it is desirable to minimize the pipeline granularity to reduce processing latency and memory sizes.

The largest CTU needs 6 kB to store its luma and chroma pixels with 8-bit precision. The transform coefficients and residue are computed with higher precision (16-bit and 9-bit, respectively) and require larger storage accordingly. Other information such as intra-prediction mode, inter-prediction motion vectors, etc. needs to be stored at a 4×4 granularity. All of these require large pipeline buffers in SRAM and several techniques can be used to reduce their size as described in this chapter.

Line buffers are required to handle data dependencies between CTUs in the vertical direction. For example, the deblocking filter needs to store four rows of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00314

Table 10.1 CTU-adaptive pipeline granularity

Coding Tree Unit (CTU)	Variable-sized Pipeline Block (VPB)
64 × 64	64 × 64
32 × 32	64 × 32
16 × 16	64 × 16

luma pixels and two rows of chroma pixels (per chroma component) due to the deblocking filter's support. The size of these buffers is proportional to the width of the picture. Further, if the picture is split into multiple tile rows, each tile row needs a separate line buffer if the rows are to be processed in parallel. Tiles also need column buffers to handle data dependencies between them in the horizontal direction. Traditionally, line buffers have been implemented using on-chip SRAM. However, for very large picture sizes, it may be necessary to store them in the denser off-chip DRAM. This results in an area and power trade-off as communicating to the off-chip DRAM takes much more power.

Also, off-chip DRAM is used most commonly to store the decoded picture buffer. The variable latency to the off-chip DRAM must be considered in the system pipeline. In particular, buffers are needed between processing blocks that talk to the DRAM to accommodate the variable latency. Motion compensation makes the most number of accesses to the external DRAM and a motion compensation cache is typically used to reduce the number of accesses. With a cache, the best-case latency for a memory access is determined by a cache hit and it can be as low as one cycle. However, the worse-case latency, determined by a cache miss, remains more or less unchanged thus increasing the overall variability seen by the prediction block.

To summarize, the top-level system pipeline is affected by:

1. Processing dependencies
2. Large CTU sizes
3. Large line buffers
4. Off-chip DRAM latency

10.2.1 Variable-Sized Pipeline Blocks

Compared to the all-intra or all-inter macroblocks in H.264/AVC, the Coding Tree Units (CTU) in HEVC may contain a mix of inter and intra-coded Coding Units. Hence, it is convenient to design the pipeline granularity to be equal to the CTU size. If the pipeline buffers are implemented as multi-bank SRAM, the decoder can be made power-scalable for smaller CTU sizes by shutting down the unused banks. However, it is also possible to use the unused banks and increase the pipeline granularity beyond the CTU size. For example, a CTU-adaptive pipeline granularity shown in Table 10.1 is employed by [9].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00315

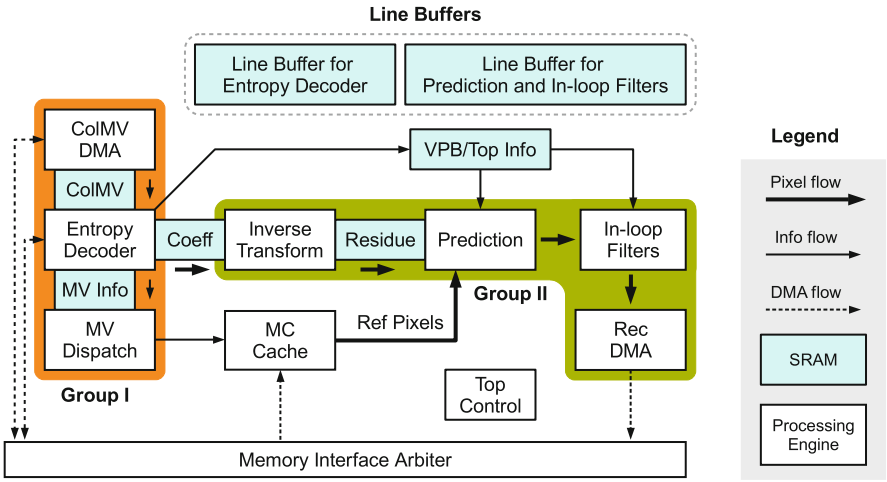


Fig. 10.1 System pipelining for HEVC decoder. Coeff buffer saves 20 kB of SRAM by TU pipelining. Connections to Line Buffers are omitted in the figure for clarity (see Fig. 10.3 for details)

The Variable-sized Pipeline Block (VPB) is as tall as the CTU but its width is fixed to 64 for a unified control flow. Also, by making the VPB larger than the CTU (for CTU 32×32 and 16×16), motion compensation can predict a larger block of luma pixels before predicting the chroma pixels. This reduces the number of switches between luma and chroma memory accesses which, as explained later in Sect. 10.6, can have benefits on the DRAM latency.

10.2.2 Split System Pipeline

To deal with the variable latency of the cache+DRAM memory system, elastic pipelining can be used between the entropy decoder, which sends read requests to the cache, and prediction, which reads data from the cache. As a result, the system pipeline can be broken into two groups. The first group contains the entropy decoder while the second contains inverse transform, prediction and the subsequent in-loop filters. This scheme is shown in Fig. 10.1.

Entropy decoder uses collocated motion vectors from decoded pictures for motion vector prediction. A separate pipeline stage, ColMV DMA is added prior to entropy decoder to read collocated motion vectors from the DRAM. This isolates entropy decoder from the variable DRAM latency. Similarly, an extra stage, reconstruction DMA, is added after the in-loop filters in the second pipeline group to write back fully reconstructed pixels to DRAM. Processing engines are pipelined with VPB granularity *within* each group as shown in Fig. 10.2. Pipelining *across* the groups is explained next.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00316

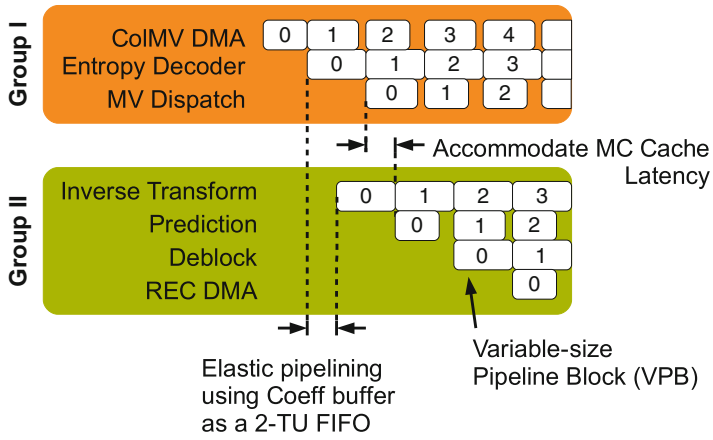


Fig. 10.2 Split system pipeline to address variable DRAM latency. Within each group, variable-sized pipeline block-level pipelining is used

The entropy decoder must send residue coefficients and transform information such as quantization parameter and TU size to the inverse transform block. As residue coefficients use 16-bit precision, 12 kB of SRAM is needed for luma and chroma coefficients of one VPB. For full pipelining, storage for two VPBs is needed so that entropy decoder can write coefficients and inverse transform can read coefficients of the previous VPB simultaneously. Thus, VPB pipelining would need 24 kB of SRAM. But this can be avoided by using the fact that the largest TU size is 32×32 (a 64×64 CU must split its transform quadtree at least once). Hence, it is possible to use a 2-TU buffer instead. The entropy decoder writes to one TU while inverse transform reads from the previous TU. This buffer requires only 4 kB, thus saving 20 kB of SRAM.

In the first pipeline group, a line buffer is used by entropy decoder for storing prediction information of upper row VPBs. In the second pipeline group, the 9-bit residues are passed from inverse transform to prediction using two VPB-sized SRAMs in ping-pong configuration. (Inverse transform writes one VPB to one SRAM while prediction reads the previous VPB from the other SRAM. When both modules are finished processing their respective VPBs, the two SRAMs switch roles.) Prediction, in-loop filters and reconstruction DMA communicate using three VPB-sized SRAMs in a rotating buffer configuration as shown in Fig. 10.3. Another line buffer is used to communicate pixels and parameters across VPB rows. The line buffer must store:

- four luma and two chroma rows (pre-deblocking) for deblocking filter. Of these, one luma and one chroma rows are also used as top neighbor pixels for intra prediction.
- one luma and one chroma rows (post-deblocking) for SAO filter

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00317

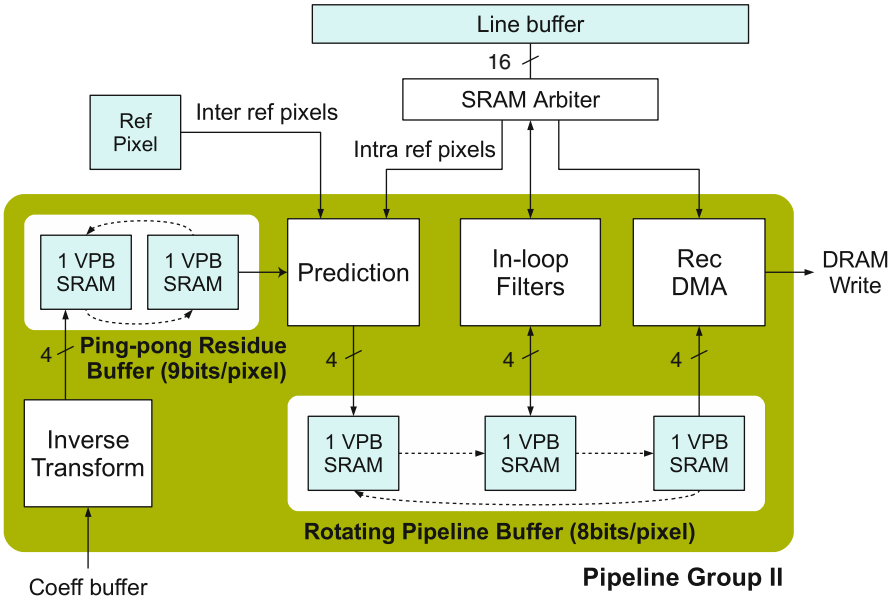


Fig. 10.3 Memory management in second pipeline group. A 2-VPB ping-pong and a 3-VPB rotating buffer are used as pipeline buffers. A single-port SRAM is used for pixel linebuffer to save area and access to it is arbitrated. Marked bus widths denote SRAM data width in pixels

- Prediction and transform parameters such as prediction mode, motion vectors, reference picture indices, intra-prediction mode and quantization parameter to determine deblocking filter parameters
- SAO parameters

To reduce the area of the line buffer, a single-port SRAM is used and requests from prediction, in-loop filters and reconstruction DMA are arbitrated. The access patterns of the three modules to the SRAM are designed to minimize the amount of collisions and the arbitration scheme gives higher priority to the deblocking filter as it has a lower margin in the cycle budget. This minimizes the performance penalty of the SRAM sharing.

10.3 Entropy Decoding

HEVC uses a form of entropy coding called Context Adaptive Binary Arithmetic Coding (CABAC) to perform lossless compression on the syntax elements [13]. Figure 10.4 shows the top level architecture of a CABAC entropy decoder. The arithmetic decoder decompresses the bitstream to generate a sequence of binary symbols (bins). The context selection finite-state-machine (FSM) determines which

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00318

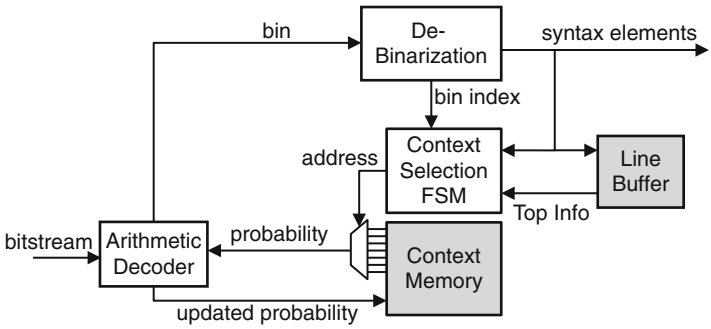


Fig. 10.4 Top-level architecture for CABAC. Memories are shown with *grey boxes*

probability should be read from the context memory based on the type of the syntax element being processed, as well as the bin index, neighboring information (top neighbor is read from a line buffer), and component (i.e., luma or chroma). When the probability used to decode a bin is read from the context memory, it is referred to as a regular coded bin; otherwise, a probability of 0.5 is assumed and the bin is referred to as bypass coded. Bypass coded bins can be decoded much faster than regular coded bins. After each regular coded bin is decoded, an updated context with the updated probability estimate is sent back to the context memory. Finally, the debinarization module maps the sequence of bins to a syntax element.

The CABAC in HEVC was redesigned for higher throughput [17]. Specifically, the CABAC in HEVC has fewer regular coded bins compared to H.264/AVC. In addition, the context selection FSM is simplified by reducing dependencies across consecutive states. Both the line buffer and context memory sizes are reduced. The number of types of binarization has increased in order to account for the reduction in regular coded bins, without coding loss. More details on this can be found in Chap. 8. HEVC uses the same arithmetic decoder as H.264/AVC.

10.3.1 Implementation Challenges

The challenge with CABAC is that it inherently has a different workload than the rest of the decoder. The workload of the entropy decoder varies based on bit-rate, while the rest of the decoder varies based on pixel-rate. The workload of CABAC can vary widely per block of pixels (i.e. CTU). Thus a high throughput CABAC is needed to order to handle the peaks of the workload variation to prevent stalls in the decoder pipeline. However, it is difficult to parallelize the CABAC due to its strong data dependencies. This is particularly true at the decoder where the data dependencies result in feedback loops. For H.264/AVC CABAC, decoders have throughput on the order of hundreds of Mbin/s versus up to Gbin/s for encoders.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00319

10.3.2 Solutions

There are several approaches that have been explored to increase the throughput of CABAC, which is dictated by the number of binary symbols it can decode per second (bin-rate). One method is to pipeline the CABAC to reduce the critical path delay [26]. However, the deeper the pipeline, the more stalls or more speculative computations/branching required. Alternatively, multiple arithmetic decoders are concatenated to decode multiple bins per cycle [12, 25]. As the number of bins per cycle increases, the number of speculative computations increases exponentially and the critical path delay increases linearly. Finally, another approach is to decode a variable number of bins per cycle, and assume that the most probable bins are decoded each cycle [27]. As the number of bins increases, the number of speculative computations only increases linearly; however, the critical path delay also increases linearly and the number of bins decoded per cycle increases less than linearly, which results in lower bin-rate. More discussion on this can be found in [16]. To address these challenges, the CABAC in HEVC minimizes dependencies across consecutive bins, particularly for the residual coding, and has fewer regular coded bins in order to reduce the amount of speculative computation required when using the pipelining or multiple bins architectures. In addition, it also groups bypass bins to enable the decoder to fully leverage the fast decoding of bypass coded bins [18].

To address the imbalance in workload between entropy decoding (Group I in Fig. 10.2) and the rest of the decoder (Group II in Fig. 10.2), a very large buffer can be inserted after the entropy decoder to average out the workload. Note that the standard constrains the workload of the entropy decoder at the frame level (using max *BinCountsInNalUnits*) and across frames (using max bit-rate in the level limit); thus using frame level buffering between the entropy decoder and the rest of the decoder can help to address this imbalance. This is commonly referred to as entropy decoupling. However, this comes at the cost of an additional frame delay and increased memory bandwidth. The memory bandwidth cost can be reduced if the intermediate values are stored as binary symbols of the CABAC rather than the reconstructed syntax elements [10]. An added advantage of having frame level buffering is that multiple rows of CTU can be decoded in parallel, since all the decoded syntax elements for the frame can be read from the buffer [5]. These new features enable CABAC decoders in HEVC to achieve higher throughput at lower cost than in H,264.AVC as demonstrated in [31].

If latency cannot be tolerated, HEVC contains high level parallelism tools such as slices, tiles and wavefront parallel processing, which enable multiple CABAC decoders to operate in parallel on the same frame. However, there is no guarantee that these features will be enabled by the encoder.

10.4 Inverse Transform and Dequantization

Dequantization scales up coefficients decoded by the entropy decoder and inverse transform converts the scaled coefficients to residue pixels using a 2-D Inverse Discrete Cosine Transform (IDCT) or a 2-D Inverse Discrete Sine Transform

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00320

(IDST). As compared to H.264/AVC, the HEVC inverse transform involves significant challenges for hardware implementation. This is the result of the following factors:

1. HEVC uses Transform Units (TUs) of size 4×4, 8×8, 16×16, and 32×32 pixels. This variety of TU sizes complicates the design of control logic as TUs of different sizes take different number of cycles for processing.
2. Like H.264/AVC, the 2-D transforms in HEVC are separable into 1-D transforms along the columns and rows. An $N \times N$ 2-D transform consists of N 1-D column transforms and N 1-D row transforms, each of which can be viewed as the product of an $N \times N$ transform matrix with $N \times 1$ input coefficients. The total number of multiplications is thus, $2N^3$ or $2N$ per coefficient. Hence, the largest IDCT in HEVC (32×32) takes $4\times$ the number of multiplications per coefficient as compared to the largest IDCT in H.264/AVC (8×8). Furthermore, the increased precision in HEVC transforms doubles the cost of each multiplication. Hence, HEVC transform logic has $8\times$ the computational complexity of H.264/AVC.
3. An intermediate memory is needed to store the TU between the column and row transforms operation. This memory must perform a transposition (i.e. columns are written to it and rows are read out). Previous designs for H.264/AVC used register arrays due to the small TU sizes. These do not scale very well to the higher TU sizes of HEVC and one must look to denser memories such as SRAM to achieve an area-efficient implementation. However, the higher density of SRAMs comes at the cost of lower memory throughput and less flexibility in read-write patterns.

A single-cycle 32-pt 1-D IDCT with Booth encoded shift-and-add multipliers takes about 145 kgate of logic. For comparison, a complete 1080p H.264/AVC decoder can be built in 160 kgate [11]. Hence, aggressive optimizations that exploit various properties of the transform matrix are necessary to achieve a reasonable area. Also, a single-cycle 32-pt IDCT provides much higher throughput than what is required for real-time operation. It is possible to reduce the area by computing the DCT over multiple cycles using partial matrix multiplication. A 2 pixel/cycle throughput at 200 MHz is sufficient for 4K Ultra HD decode at 30 fps. The following subsections describe such a design.

10.4.1 Top-Level Pipelining

In general, two high-level architectures are possible for a 2 pixel/cycle inverse transform [4]. The first one, shown in Fig. 10.5a uses separate stages for row and column transforms. Each one has a throughput of 2 pixel/cycle and operates concurrently. The dependency between the row and column transforms (all columns of the TU must be processed before the row transform) means that the two stages must process different TUs at the same time. The transpose memory must have one

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00321

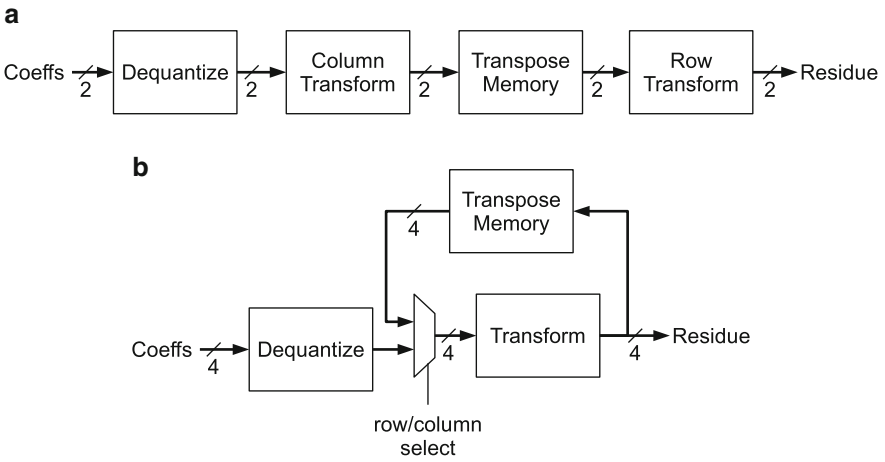


Fig. 10.5 Possible high-level architectures for inverse transform with 2 pixel/cycle throughput. Bus-widths are in pixels. (**a**) Separate row and column transform stages. (**b**) 1-D transform stage shared by row and column transform

read and one write port and hold two TUs—in the worst case, two 32 × 32 TUs. Also, the two TUs would take different number of cycles to finish processing. For example, if a 8×8 TU follows a 16×16 TU, the column transform must remain idle after processing the smaller TU as it waits for the row transform to finish the larger one. It can begin processing the next TU but managing several TUs in the pipeline at the same time will require complex control logic to avoid stalls.

With these considerations, the second architecture, shown in Fig. 10.5b is preferred. This uses a single 4 pixel/cycle 1-D transform for both row and column transform to achieve the desired 2 pixel/cycle 2-D transform throughput. The 1-D transform works on a single TU at a time, processing all the columns first and then all the rows. Hence, the transpose memory needs to hold only one TU and can be implemented with a single port SRAM since row and column transforms do not occur concurrently.

10.4.2 Transpose Memory

The transform block uses a 16-bit precision input for both row and column transforms. The transpose memory must be sized for 32 × 32 TU which means a total size of $16 \times 32 \times 32 = 16.4$ kbit. In comparison, H.264/AVC decoder designs require a much smaller transpose memory—$16 \times 8 \times 8 = 1$ kbit. A 16.4 kbit memory

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00322

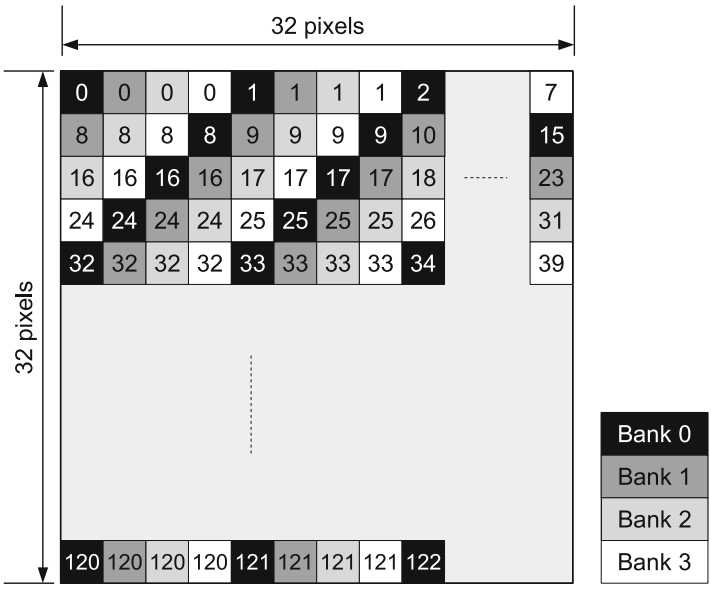


Fig. 10.6 Mapping a 32 × 32 TU to four SRAM banks for transpose operation. The *color* of each pixel denotes the bank and the *number* denotes the bank address

with the necessary read circuit for the transpose operation takes up a lot of area (125 kgate) when implemented with registers and multiplexers. Also, the register-based transpose memory has a much higher throughput than required. SRAMs are more area-efficient than registers and have a lower throughput, which makes them a good choice for an optimized implementation. The main disadvantage of SRAMs is that they are less flexible than registers. A register array allows reading and writing to arbitrary number of bits at arbitrary locations, although very complicated read(write) patterns would lead to a large output(input) mux size. The SRAM read or write operation is limited by the bit-width of its port. A single-port SRAM allows only one operation, read or write, every cycle. Adding extra ports is possible at the expense of significant area increase.

It is possible to implement the 4-pixel/cycle transpose memory using four single-port banks of 4,096 bits each with a port-width of 1 pixel. The pixels in a 32×32 TU are mapped to locations in the four banks as shown in Fig. 10.6. By ensuring that four adjacent pixels in any row or column sit in different SRAM banks, it is possible to write along columns and read along rows by supplying different addresses to the four banks.

After a 32-pt column transform is computed, the result is saved in a temporary register and is written to the transpose SRAM over eight cycles. At the same time, the 1-D transform module processes the next column. This is shown in cycles 0–7 in Fig. 10.7a, where the result of column 30 is written to the SRAM while the 1-D transform module works on column 31. However, when the last column in a TU is

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00323

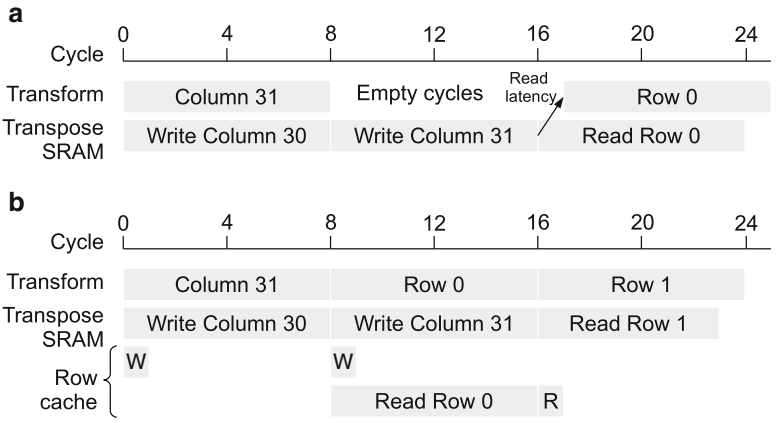


Fig. 10.7 Eliminate read/write with registers for an SRAM-based transpose memory. (**a**) Pipeline stall due to transpose SRAM delay for 32 × 32 TU. (**b**) Row caching to avoid stall

processed, the transform module must wait for it to be written to the SRAM before it can begin processing the row. This results in a delay of nine cycles for 32 × 32 TU. In general, for an N×N TU, this delay is equal to N/4 + 1 cycles. This results in a pipeline stall of 1.75–25 % cycles depending on the TU size. This stall can be avoided through the use of a row cache that stores the first N + 4 pixels in registers. As shown in Fig. 10.7b, the row cache is read for the first nine cycles of the row transforms while the last column is being stored in the SRAM.

This transpose memory design using SRAM scales very well for lower throughputs. A 2-pixel/cycle transpose memory would need two banks each with 512 entries (16-bit/entry). For higher throughputs, one needs more banks each with fewer entries. Such short SRAM banks have a larger area overhead of sense-amplifiers and other read-out circuitry. For throughputs higher than 32-pixel/cycle, register based transpose memory [23] is more area-efficient.

10.4.3 Inverse DCT Engine

The IDCT engine can be optimized by observing that the N-pt IDCT matrix has at most N unique coefficients differing only in sign. This is also true of the matrices obtained by even-odd decomposition of the IDCT matrix, such as the 16 × 16 matrix of the 32-pt IDCT. This 256-element matrix contains 15 unique numbers: 90, 88, 85, 82, 78, 73, 67, 61, 54, 46, 38, 31, 22, 13, 4 (and their additive inverses). The matrix is multiplied with the odd-indexed coefficients in the 32-pt IDCT. In a 4-pixel/cycle

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00324

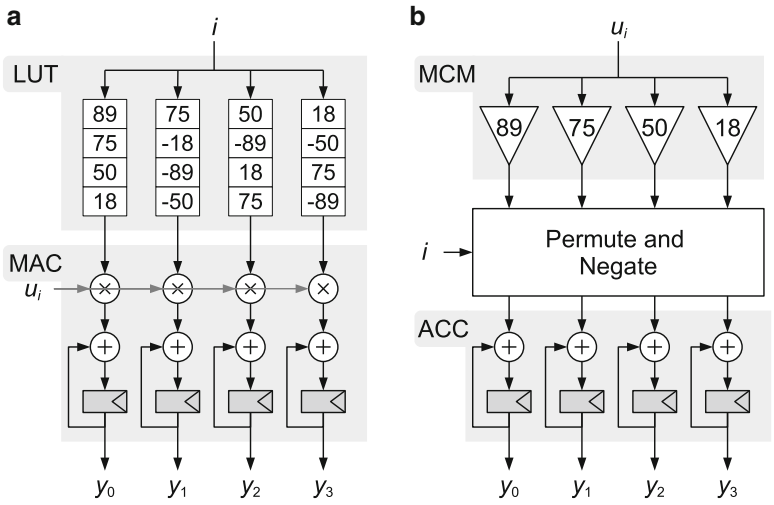


Fig. 10.8 4×4 matrix multiplication in Eq. (10.1) without and with unique operations. (**a**) Generic implementation. (**b**) Exploiting unique operations

Table 10.2 Area reduction by exploiting unique operations

Matrix multiplication	Area for generic implementation (kgates)	Area exploiting unique operations (kgates)	Area savings
4 × 4	10.7	7.3	32 %
8 × 8	23.2	13.5	42 %
16 × 16	46.7	34.4	26 %

case, only two of these inputs are available per cycle. So, it is enough to perform a partial 2 × 16 matrix multiplication every cycle and accumulate the outputs over eight cycles. In general, this would require 32 full multipliers and 32 lookup tables to store the matrix. However, knowing that the matrix has only 15 unique numbers, we can simply instantiate 15 constant multipliers with some negators and multiplexers to implement the matrix multiplication. This is shown for the 4 × 4 odd matrix multiplication (Eq. (10.1)) of the 8-pt IDCT in Fig. 10.8b. The area savings are shown in Table 10.2.

$$\begin{bmatrix} y_0 & y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} 89 & 75 & 50 & 18 \\ 75 & -18 & -89 & -50 \\ 50 & -89 & 18 & 75 \\ 18 & -50 & 75 & -89 \end{bmatrix} \tag{10.1}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00325

Table 10.3 Area breakdown for inverse transform

Module	Logic area (kgates)
Partial transform	71
Accumulator	5
Row cache	4
FIFOs	5
Scaling + Control	19
Total	104

Table 10.4 Area for different transforms. Partial 32-pt IDCT contains all the smaller IDCTs

Module	Logic area (kgates)
4-pt IDCT	3
Partial 8-pt IDCT	10
Partial 16-pt IDCT	24
Partial 32-pt IDCT	57
4-pt IDST + misc.	14

10.4.4 Implementation Results

Breakdown of the post-synthesis logic area at 200 MHz clock frequency in 40 nm CMOS is given in Table 10.3. The total area is 104 kgate of logic (in terms of 2-input NAND gates) and 16.4 kbit of SRAM. Table 10.4 shows the combinational logic area required for 1-D transform operations. Data-gating and zero-column skipping can provide power reduction of 18% and throughput improvement of 27-66% as shown in [32].

10.5 Inter Prediction

HEVC inter prediction uses motion vectors pointing to one reference frame (uni-prediction) or two reference frames (bi-prediction) to predict a block of pixels. The size of the predicted block, called Prediction Unit (PU), is determined by the Coding Unit (CU) size and its partitioning mode. For example, a 32 × 32 CU with 2N×N partitioning is split into two PUs of size 32 × 16, or a 16 × 16 CU with nL×2N partitioning is split into 4 × 16 and 12 × 16 PUs.

For luma pixels, the motion vectors for each PU have a resolution of 1/4-th pixel. The predicted pixels at non-integer pixel positions are obtained by interpolating between the reference pixels using an 8-tap FIR filter, first along the horizontal direction and then along the vertical as shown in Fig. 10.9. (In Main Profile, the reverse order, i.e. vertical followed by horizontal also gives the same result). For chroma, the motion vector is halved and has a 1/8-th pixel resolution computed using a 4-tap interpolation filter. From Table 10.5, which shows the cost of interpolating

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00326

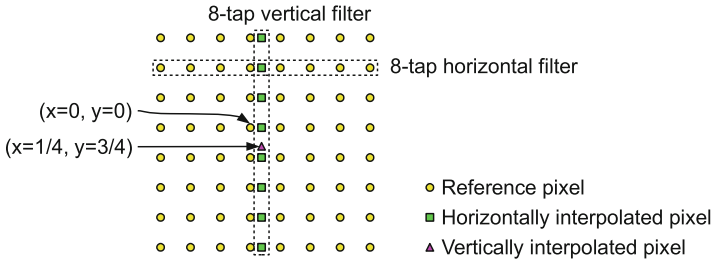


Fig. 10.9 Interpolation process for a pixel at a fractional location $x = 1/4, y = 3/4$

Table 10.5 Example costs for interpolating a block of pixels

	Block type	Generic	Y64 × 64	Y16 × 16	U4x4
Parameters	Block size	$w \times h$	64 × 64	16 × 16	4 × 4
	Filter size	$n + 1$ taps	8 taps	8 taps	4 taps
Costs	Reference pixels	$(w + n) \times (h + n)$	71 × 71 (23 %)	23 × 23 (106 %)	7 × 7 (206 %)
	Horizontal interps.	$w \times (h + n)$	64 × 71 (11 %)	16 × 23 (43 %)	4 × 7 (75 %)
	Vertical interps.	$w \times h$	64 × 64 (0 %)	16 × 16 (0 %)	8 × 8 (0 %)

Values in brackets denote overhead over the block size. Costs are for uni-prediction only. For bi-prediction, all the costs are doubled

a block of pixels, we see that smaller pixel blocks have a proportionately higher overhead in the number of reference pixels and number of horizontal interpolations. To reduce the worst case overhead, 4 × 4 PUs are not allowed by the standard and 8 × 4/4 × 8 PUs are allowed to use only uni-prediction.

Compared to H.264/AVC, HEVC uses

1. Larger PUs which require fewer interpolations per pixel but more on-chip SRAM
2. More varied PU sizes which increase complexity of control logic
3. Longer interpolation filters which require more datapath logic and more reference pixels

Reference frames may be stored in off-chip DRAM for HD and larger picture sizes, or in on-chip SRAM for smaller sizes. At a PU level, it is observed that reference pixels of adjacent PUs have significant overlap. Due to this spatial locality, fetching the reference pixels into a motion-compensation (MC) cache helps reduce the latency and power required to access external DRAM and large on-chip SRAMs. Considering this, a top-level architecture (showing only the data-path) for an HEVC inter-prediction engine would look like Fig. 10.10.

The Dispatch module generates the position and size of the reference pixel block according to the decoded motion vectors (MVs). The MC Cache will send read requests to reference frame buffer over the direct-memory-access (DMA) bus for cache misses. When all the reference pixels are present in the MC cache, the Fetch module will fetch them from the cache for the 2-D Filter module. Note that it could take many cycles to get data from DMA bus, due to latencies of bus arbiters, DRAM controller, and DRAM Precharge/Activate operations.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00327

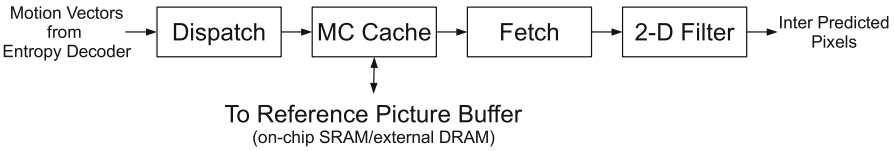


Fig. 10.10 System architecture for HEVC inter prediction. Only main data flow is shown

The following subsections describe techniques used to address the important challenges of implementing HEVC inter prediction in hardware.

1. A fixed pipelining across the Dispatch, Fetch and 2-D Filter modules for simpler control and reduced on-chip SRAM
2. A PU-adaptive scheduling within each module to handle the variety of PU sizes
3. Time-multiplexed Multiple Constant Multiplication (TMMCM) [21] to reduce interpolation filter size

Section 10.6 describes the design of a motion compensation cache used to reduce the memory bandwidth requirement and power consumption of the reference picture buffer.

10.5.1 Fixed Pipelining Across Modules

In HEVC, it is possible to predict a large block of pixels in smaller pipeline blocks by treating the smaller blocks as independent PUs with the same motion vector information. So, to deal with all the variety of PU sizes, one can use a constant block size of 4×4. This drastically reduces the size of pipeline buffers between the modules in Fig. 10.10. However, as explained previously, the smaller blocks have a larger overhead in terms of fetching reference pixels and performing horizontal interpolations.

In [33], 16×16 pipeline blocks are used to tradeoff SRAM size and computation overhead. For chroma, since a block of 16×16 luma pixels corresponds to two 8×8 chroma pixels in the 4:2:0 format, chroma pixels from two 16×16 blocks are combined and used as a single pipeline block of four 8×8 pixels. As compared to a 64×64 CTU granularity, this requires 24× smaller pipeline buffers. The worst case overhead of this scheme is seen when a 64×64 PU is split into 16×16 pipeline blocks. For luma pixels, this PU originally requires $64 \times 71 = 4{,}544$ horizontal interpolations but processing it in smaller blocks increases that by 30 % to $16 \times (16 \times 23) = 5{,}888$. For PU sizes smaller than 16×16, multiple such PUs are combined into one pipeline block.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00328

Table 10.6 Number of horizontal interpolations for each PU type

PU Type	Uni/bi directional	No. of horizontal interpolations		No. of vertical interpolations	
		per PU	per pixel	per PU	per pixel
Y16 × 16	Uni/bi	2 × 16 × 23	2.875	2 × 16 × 16	2
Y8 × 8	Uni/bi	2 × 8 × 15	3.75	2 × 8 × 8	2
Y16 × 4	Uni/bi	2 × 16 × 11	5.5	2 × 16 × 4	2
Y4 × 16	Uni/bi	2 × 4 × 23	2.875	2 × 4 × 16	2
Y8 × 4	Uni	8 × 11	2.75	8 × 4	1
Y4 × 8	Uni	4 × 15	1.875	4 × 8	1
UV8 × 8	Uni/bi	2 × 8 × 11	2.75	2 × 8 × 8	2
UV4 × 4	Uni/bi	2 × 4 × 7	3.5	2 × 4 × 4	2
UV8 × 2	Uni/bi	2 × 8 × 5	5	2 × 8 × 2	2
UV2 × 8	Uni/bi	2 × 2 × 11	2.75	2 × 2 × 8	2
UV4 × 2	Uni	4 × 5	2.5	4 × 2	1
UV2 × 4	Uni	2 × 7	1.75	2 × 4	1

Some PU types are restricted to uni-prediction while other types can use either

10.5.2 PU-Adaptive Pipelining in 2-D Filter

The 2-D Filter must handle PUs of size 16 × 16 and smaller for luma and chroma which require different number of interpolations as shown in Table 10.6. Y16 × 4 PU requires the most number of horizontal interpolations (5.5 per pixel) and so, for a 2 pixel/cycle throughput, 11 horizontal filters are required. By a similar analysis, four vertical filters are required. However, this would result in a mismatch between the peak throughput of the horizontal filters (11 pixel/cycle) and the vertical filters. The designer can choose to add a buffer after the horizontal filters to handle the mismatch or match the peak throughput with 11 vertical filters.

10.5.3 TMMCM for Interpolation Filter

The 6-tap interpolation filter in H.264/AVC is easy to optimize due to its symmetry and simple coefficients [1]. However, HEVC uses longer 8-tap and 4-tap filters for luma and chroma coefficients respectively, and the filter coefficients are also more complex. In [6], a 1-D luma filter design with 16 adders and a 2-D filter reuse scheme for sub-block 4 × 4 are proposed. A 1-D filter design using only 13 adders is also possible by unifying the luma and chroma filters into one single design and optimizing it with time-multiplexed multiple-constant multiplication (TMMCM). TMMCM is similar to MCM seen in Sect. 10.4 on Inverse Transform. However, exactly one of the MCM outputs is needed every clock cycle and this allows further optimizations by placing multiplexers within the MCM adder tree. One such TMMCM optimization is explained in some detail next.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00329

Selection	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Y 0, U 0				64				
Y 1/4, 3/4	-1	4	-10	58	17	-5	1	0
Y 1/2	-1	4	-11	40	40	-11	4	-1
UV 1/8, 7/8		-2		58	10		-2	
UV 1/4, 3/4		-4		54	16		-2	
UV 3/8, 5/8		-6		46	28		-4	
UV 1/2		-4		36	36		-4	

Fig. 10.11 Unified luma and chroma interpolation filters with inputs reordered. The coefficients for x_3 (in *dashed box*) can be implemented with two adders and three multiplexers as shown in Fig. 10.12

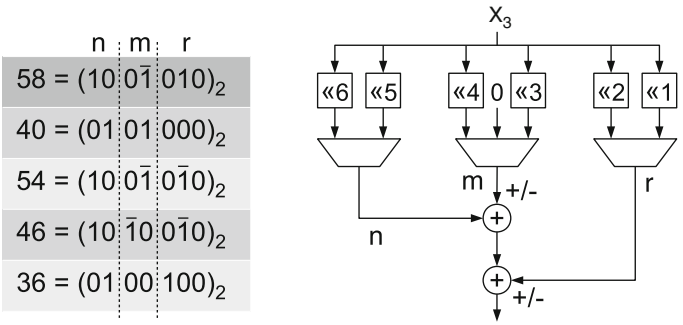


Fig. 10.12 Time-multiplexed Multiple Constant Multiplication for x_3

A reorder of the filter inputs is first applied to reduce complexity based on symmetry as shown in Fig. 10.11. Note that two sets of the chroma filter coefficients are placed in x_1 and x_6, instead of x_2 and x_5, due to the similarity with the luma coefficients 4 and 1. There are only seven cases left. The design principle adopted here is to optimize TMMCM coefficients for each filter input. As an example, the design for x_3 is shown in Fig. 10.12.

In the canonical signed digit representation, the coefficients have at most three non-zero digits which determines the number of adders to be 2. The non-zero digits are partitioned into three groups (n, m and r) such that each group has at most one non-zero digit. Finally, the three partitions are summed with partitions having similar bitwidths added first.

Compared to algorithmically generated filter designs using [15], this design has a 5–31 % lower area as shown in Table 10.7.

Combining all the presented techniques, the complete 1-D filter is shown in Fig. 10.13 using only 13 adders. Regarding the bitwidth increase between the input and output, the case of luma 1/2-pel position gives the largest values for both

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00330

Table 10.7 Gate counts of the described and reference designs for the x_3, x_4, and $x_2|x_5$ TMMCM in the vertical filter based on 40 nm process synthesis results

Design	x_3 TMMCM		x_4 TMMCM		$x_2\|x_5$	
Timing	1 ns	2 ns	1 ns	2 ns	1 ns	2 ns
Reference (gates)	1144	547	557	526	2284	845
Proposed (gate)	1036	518	442	361	1578	738
Area reduction	9.4 %	5.4 %	20.6 %	31.4 %	30.9 %	12.6 %

The reference designs for x_3 and x_4 are generated by [15], and the reference for $x_2|x_5$ is designing x_2 and x_5 separately

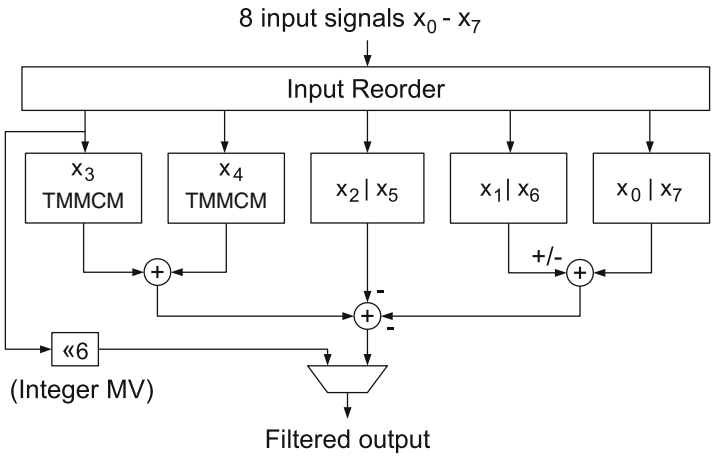


Fig. 10.13 HEVC interpolation filter design using 13 adders

unsigned and signed inputs, and the outputs can be magnified at most by 88 and 112 times respectively. So, the 1-D horizontal filter has 8-bit unsigned input and 16-bit signed output, and the vertical one has 16-bit signed input and 23-bit signed output.

10.5.4 Implementation Results

For supporting 4K Ultra-HD 30 fps videos, this architecture is synthesized at 200 MHz in 40 nm CMOS. The result is shown in Table 10.8. The total gate count is 69.4k, of which 50.0k for the 2-D filter. The Fetch module mainly consists of large multiplexers and results in 12.0 kgate. The Dispatch module occupies 4.7 kgate for the block size and position calculation. The total SRAM size is 31 kbit, including the two-port 2.2 kbit Dispatch Info SRAM and the single-port 28.8 kbit Reference Data SRAM.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00331

Table 10.8 Gate count of inter architecture when synthesized at 200 MHz in 40 nm CMOS. SRAM sizes are also summarized

Module	Logic area (kgates)	SRAM (kbit)
Dispatch	4.7	2.2 (two-port)
Fetch	12.0	28.8 (one-port)
2-D Filter	50.0	n/a
Inter Ctrl	2.7	n/a
Total	69.4	31.0

Table 10.9 Gate count breakdown for the 2-D filter

Sub-module	Logic area (kgates)
Input Mux	4.8
H Filter	12.0
V Filter	21.8
Register Chain	9.4
Bi-Sum	1.2
Ctrl	0.8
Total	50.0

Since most of the gates are for the 2-D filter, its gate count is given in more detail in Table 10.9. For the 2-D filter, the horizontal and vertical filters occupy the most, and the area of horizontal ones is nearly one half of that of vertical ones due to their smaller internal bitwidth. This implementation does not include all PU Types used in the HEVC standard (Asymmetric Motion Partitions 32×8, 8×32, 16×4, 4×16 are not implemented), and so, uses only eight horizontal and eight vertical filters.

10.6 MC Cache and DRAM Mapping

HEVC's longer interpolation filters cause a significant increase in the required motion compensation (MC) bandwidth to the reference picture buffer (a.k.a. decoded picture buffer—DPB) as compared to H.264/AVC. However, there is significant overlap in the reference pixel data required by neighboring inter PUs which can be exploited by a cache. Most video codecs use DRAM based memory to store the DPB since it can be several megabytes large. In such a scenario, in addition to reducing the bandwidth requirement, the cache also hides the variable latency of the DRAM. This section describes the design of a read-only MC cache to support real-time decoding of 4K Ultra-HD HEVC video.

The target DRAM system is intended to store six reference pictures at 4K Ultra-HD resolution (corresponding to HEVC level 5) in addition to the collocated motion vector data. The DRAM system is composed of two 64M × 16-bit DDR3 DRAM modules with a 32 byte minimum access unit (MAU). A single MAU is mapped to a cache line.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00332

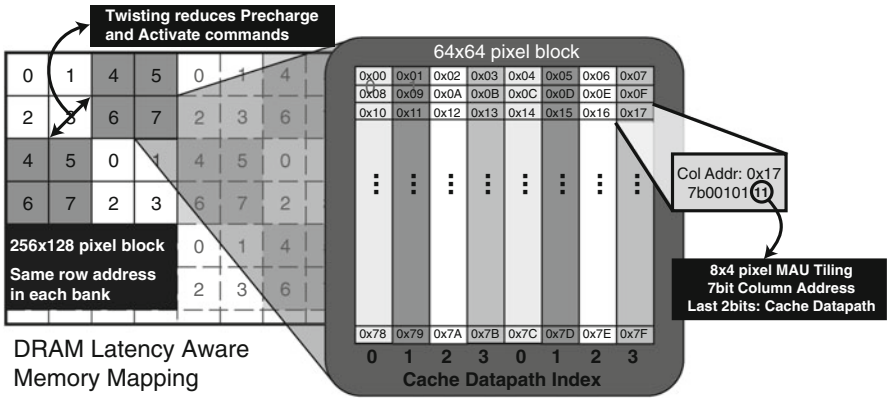


Fig. 10.14 Latency Aware DRAM mapping. 128 8 × 4 MAUs arranged in raster scan order make up one block. The twisted structure increases the horizontal distance between two rows in the same bank. Note how the MAU columns are partitioned into four datapaths (based on the last 2 bits of column address) for the four-parallel cache architecture

10.6.1 DRAM Latency Aware Memory Map

An ideal mapping of pixels to DRAM addresses should minimize the number of DRAM accesses and the latency experienced by each access. This can be achieved by minimizing the fetch of unused pixels and the number of row precharge/activate operations respectively. Note that the above optimization only fixes how the pixels are stored in DRAM and can be performed even in the absence of an MC cache. Also, the DRAM addresses should be mapped to cache lines such that conflict misses are minimized. To enable a coherent presentation, we explain these ideas with respect to a specific memory map. The underlying principles are quite general and can be easily reused.

Figure 10.14 shows an example latency aware memory map. The luma color plane of a picture is tiled by 256 × 128 pixel blocks in raster scan order. Each block maps to an entire row across all eight banks. These blocks are then broken into eight 64 × 64 blocks which map to an individual bank in each row. Within each 64 × 64 block, 32-byte MAUs map to 8 × 4 pixel blocks that are tiled in a raster scan order. In Fig. 10.14, the numbered square blocks correspond to 64 × 64 pixels and the numbers stand for the bank they belong to. Note how the mapping of 128 × 128 pixel blocks within each 256 × 128 regions alternates from left to right. Figure 10.14 shows this twisting behavior for a 128 × 128 pixel region composed of four 64 × 64 blocks that map to banks 0, 1, 2 and 3.

The chroma color plane is stored in a similar manner in different rows. The only notable difference is that an 8 × 4 chroma MAU is composed of pixel-level interleaving of 4 × 4 U and V blocks. This is done to exploit the fact that U and V have the same reference region.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00333

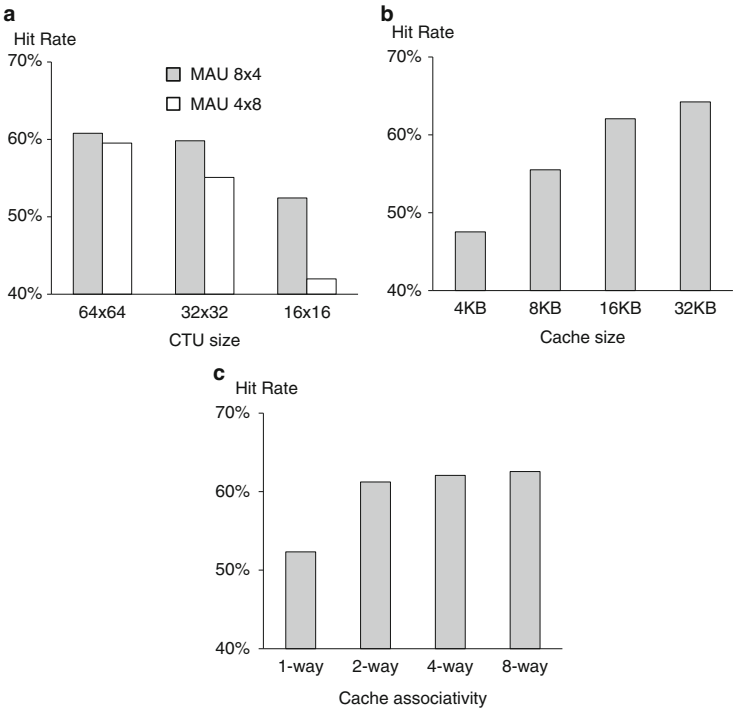


Fig. 10.15 Cache hit rate as a function of CTU size, (**a**) cache line geometry, (**b**) cache-size and (**c**) associativity. Experiments averaged over six sequences—*Basketball Drive, Park Scene, Tennis, Crowd Run, Old Town Cross* and *Park Joy*. The first are Full HD (240 pictures each) and the last three are 4K Ultra HD (120 pictures each). CTU size of 64 is used for the cache-size and associativity experiments

Minimizing Fetch of Unused Pixels. Since the MAU size is 32 bytes, each access fetches 32 pixels, some of which may not belong to the current reference region as seen in Fig. 10.16. These can be minimized by using an 8×4 MAU to exploit the rectangular geometry of the reference region. When compared with a 32×1 cache line this reduces the amount of unused pixels fetched for a given PU by 60 % on average.

Since the fetched MAU are cached, unused pixels may be reused if they fall in the reference region of a neighboring PU. Reference MAUs used for prediction at the right edge of a CTU can be reused when processing CTU to its right. However the lower CTU gets processed after an entire CTU row in the picture. Due to limited size of the cache, MAUs fetched at the bottom edge will be ejected and are not reused when predicting the lower CTU. When compared to 4×8 MAUs, 8×4 MAUs fetch more reusable pixels on the sides and less unused pixels on the bottom. As seen in Fig. 10.15a, this leads to a higher hit-rate. This effect is more pronounced for smaller CTU sizes where hit-rate may increase by up to 12 %.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00334

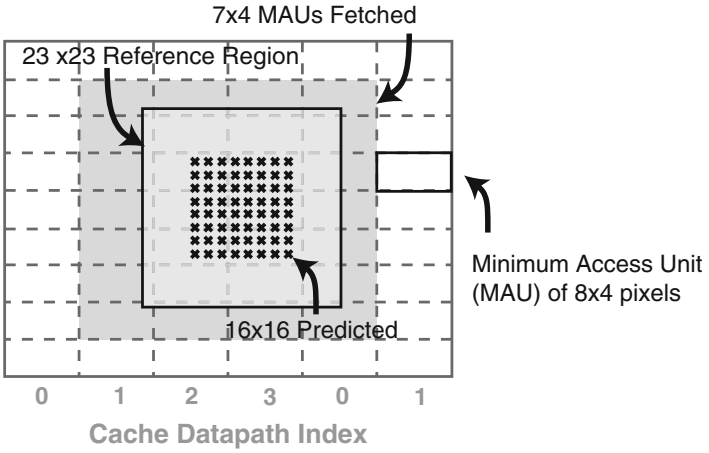


Fig. 10.16 Example of MC cache dispatch for a 23 × 23 reference region of a 16 × 16 PU. Seven cycles are required to fetch the 28 MAU at 4 MAU per cycle. Note that the dispatch region and the four parallel cache datapaths may be misaligned, thus requiring a reordering. For example, the region in this figure starts from datapath #1

Table 10.10 Comparison of twisted 2D mapping and direct 2D mapping

Encoding Mode CTU Size		LD			RA		
		64	32	16	64	32	16
ACT BW	Direct 2D	272	227	232	690	679	648
(MBytes/s)	Twisted 2D	219	183	204	667	659	636
Gain		20 %	20 %	12 %	3 %	3 %	2 %

Minimizing Row Precharge and Activation. The Twisted 2D mapping of Fig. 10.14 ensures that pixels in different DRAM rows in the same bank are at least 64 pixels away in both vertical and horizontal directions. It is unlikely that inter-prediction of two adjacent pixels will refer to two entries so far apart. Additionally a single dispatch request issued by the MC engine can at most cover four banks. It is possible to keep the corresponding rows in the four banks open and then fetch the required data. These two factors help minimize the number of row changes. Experiments show that twisting leads to a 20 % saving in bandwidth over a direct mapping as seen in Table 10.10.

Minimizing Conflict Misses. A conflict miss occurs when two locations in memory map to the same cache line. To mitigate this, we need to select an appropriate mapping between the DRAM addresses and the cache line indices. Setting the line index to the 7 bit column address of the MAU ensures that two conflicting pixel location in the same picture are at least 64 pixels apart. However, the same pixel location across two pictures will map to the same cache line. Similarly a luma and an unrelated chroma address may also map to the same cache line. Using 4-way set associativity in the cache helps resolve both these conflicts.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00335

Alternative techniques to tackle conflict misses include having separate luma and chroma caches. Similarly offsetting the memory map such that the same location in successive frames maps to different cache lines can also reduce conflicts. For our chosen configuration, the added complexity for these techniques outweighed the observed hit-rate increases.

10.6.2 Four-Parallel Cache Architecture

This section describes a four parallel MC cache architecture. Datapath parallelism and outstanding request queues for hiding the variable DRAM latency ensure a high throughput. As seen in Fig. 10.17, there are four parallel paths each outputting up to 32 pixels (1 MAU) per cycle.

10.6.2.1 Four-Parallel Data Flow

The parallelism in the cache datapath allows up to 4 MAUs in a row to be processed simultaneously. The MC cache must fetch at most 23×23 reference region corresponding to a 16×16 PU, which is the largest PU processed by Inter Prediction (see Sect. 10.5.1). This may require up to seven cycles as shown in Fig. 10.16. The address translation unit in Fig. 10.17 reorders the MAUs based on the lowest 2 bits of the column address. This maps each request to a unique datapath and allows us to split the tag register file and cache SRAM into four smaller pieces. Note that this design cannot output 2 MAUs in the same column on the same cycle. Thus our design trades unused flexibility in addressing for smaller tag-register and SRAM sizes.

The cache tags for the missed cache lines are immediately updated when the lines are requested from DRAM. This preemptive update ensures that future reads to the same cache line do not result in multiple requests to the DRAM. Note that behavior is similar to a simple non-blocking cache and does not involve any speculation. Additionally since the MC cache is a read only cache there is no need for write-back in case of eviction from the cache.

10.6.2.2 Queue Management and Hazard Control

Each datapath has independent read and write queues which help absorb the variable DRAM latency. The 32 deep read queue stores pending requests to the SRAM. The eight deep write queue stores pending cache misses which are yet to be resolved by the DRAM. The write queue is shorter because fewer cache misses are expected. Thus the cache allows for up to 32 pending requests to the DRAM. At the system level the latency of fetching the data from the DRAM is hidden by allowing for a separate motion vector (MV) dispatch stage in the pipeline prior to the Prediction

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00336

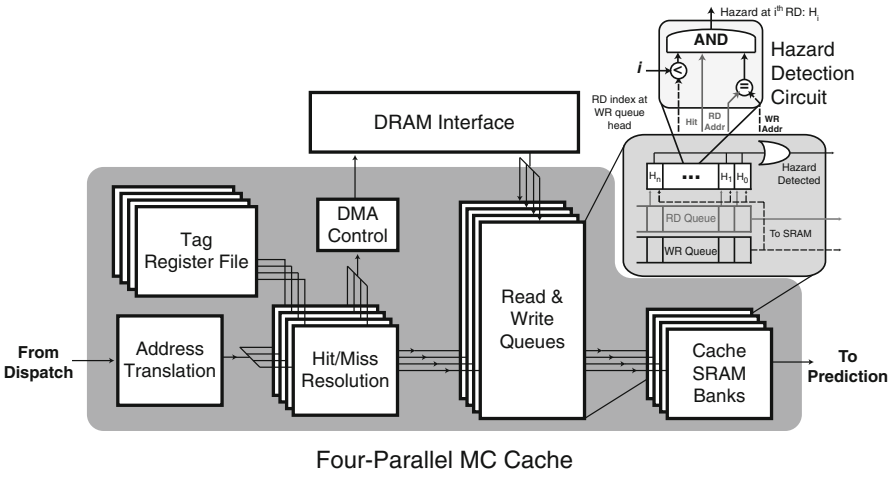


Fig. 10.17 Proposed four-parallel MC cache architecture with four independent datapaths. The hazard detection circuit is shown in detail

stage. Thus, while the reference data of a given block is being fetched, the previous block is undergoing prediction. Note that the queue sizes here are decided based on the behavior of the target DMA arbiter and DRAM latency, and for different systems they should be optimized accordingly.

Since the cache system allows multiple pending reads, write-after-read hazards are possible. For example, consider two MAUs A and B that are mapped to the same cache line. Presently, the cache line contains A, the write queue contains a pending cache miss for B and the read queue contains pending requests for A and B in that order. If B arrives from the DRAM, it must wait until A has been read from the cache to avoid evicting A before it has been read. The Hazard Detection Circuit in Fig. 10.17 detects this situation and stalls the write of B.

10.6.2.3 Cache Parameters

Figures 10.15b, c show the hit-rates observed as a function of the cache size and associativity respectively. A cache size of 16 kB was chosen since it offered a good compromise between size and cache hit-rate. The performance of FIFO replacement is as good as Least Recently Used replacement due to the relatively regular pattern of reference pixel data access. FIFO was chosen because of its simple implementation. The cache associativity of 4 is sufficient to accommodate both Random Access GOP structures and the three component planes (Y, U, V).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00337

10.6.3 Hit Rate Analysis, DRAM Bandwidth and Power

The rate at which data can be accessed from the DRAM depends on two factors: the number of bits that the DRAM interface can (theoretically) transfer per unit time and the precharge latency caused by the interaction between requests. The precharge latency can be normalized to bandwidth by multiplying with the bitwidth. This normalized figure (called ACT BW) is the bandwidth lost in the precharge and activate cycles—the amount of data that could have been transferred in the cycles when the DRAM was executing row change operation. The other figure, Data BW, refers to the amount of data that needs to be transferred from the DRAM to the decoder per unit time for real-time operation. Thus, a better hit-rate reduces the Data BW and a better memory map reduces the ACT BW. The advantage of defining Data BW and ACT BW as mentioned above is that (Data BW + ACT BW) is the minimum bandwidth required at the memory interface to support real-time operation.

The performance of the cache and the twisted address mapping is compared with two reference scenarios: raster-scan address mapping with no cache and raster scan address mapping with the cache. As seen in Fig. 10.18a, using a 16 kB cache reduces the Data BW by 55 %. The Twisted 2D mapping reduces ACT BW by 71 %. Thus, the cache results in a 67 % reduction of the total DRAM bandwidth. Using a simplified power consumption model [14] based on the number of accesses, this cache is found to save up to 112 mW, a 41 % reduction in DRAM access power as shown in Fig. 10.18b.

Figure 10.18c compares the DRAM bandwidth across various encoder settings. Smaller CTU sizes result in a larger bandwidth because of lower hit-rates. Thus, larger CTU sizes such 64 can provide smaller external bandwidth at the cost of higher on-chip complexity. Also, Random Access mode typically has lower hit rate when compared to Low Delay. This behavior is expected because the reference pictures are switched more frequently in the former.

10.6.4 Implementation Results

This design is synthesized at 200 MHz in 40 nm CMOS. The total area is 90.4 kgate of logic and 16 kB (or 131.1 kbit) of SRAM. The bulk of the logic area is taken by the 8,960 bit tag register file and can be replaced by a 2-port SRAM (which is denser than register file) at the cost of an extra access cycle. Breakdown of the logic area is presented in Table 10.11.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00338

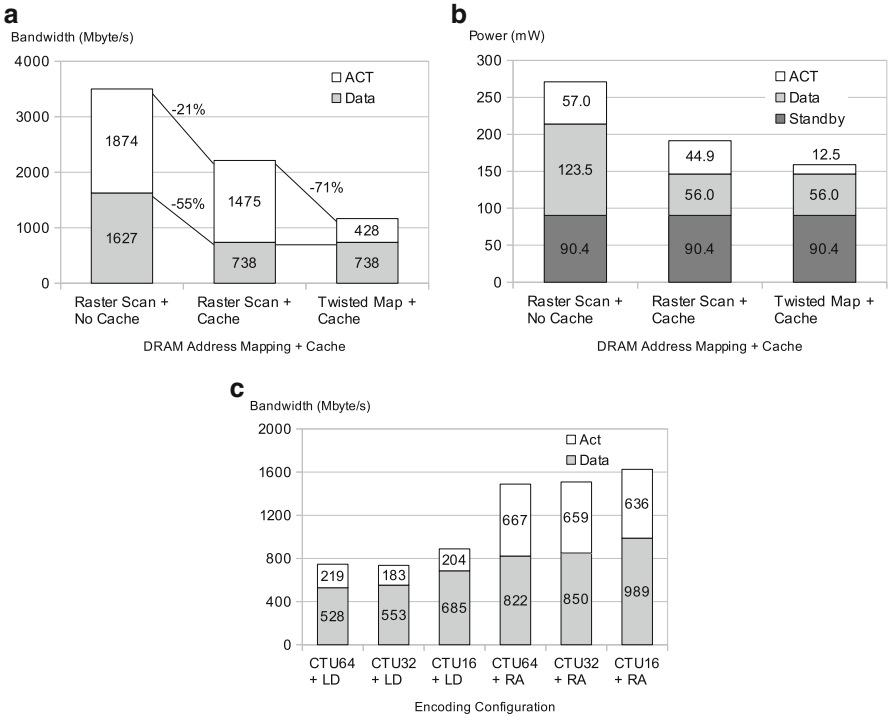


Fig. 10.18 Comparison of DDR3 bandwidth and power consumption across three scenarios. RS mapping maps all the MAUs in a raster scan order. ACT corresponds to the power and bandwidth induced by DRAM Precharge/Activate operations. (**a**) Bandwidth comparison. (**b**) Power comparison. (**c**) BW across sequences

Table 10.11 Breakdown of logic area for motion compensation cache

Module	Logic area (kgate)
Address Translation	1.1
Hit/Miss Resolution	3.9
Queue	20.5
Tag Register File	64.9
Total	90.4

10.7 Intra Prediction

Intra prediction predicts a block of pixels based on neighboring pixels in the same picture. The neighboring pixels are extrapolated into the block to be predicted along one of 33 directions or using two other intra modes—DC and Planar. The neighboring pixels are taken from one row of pixels to the top and one column to the left.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00339

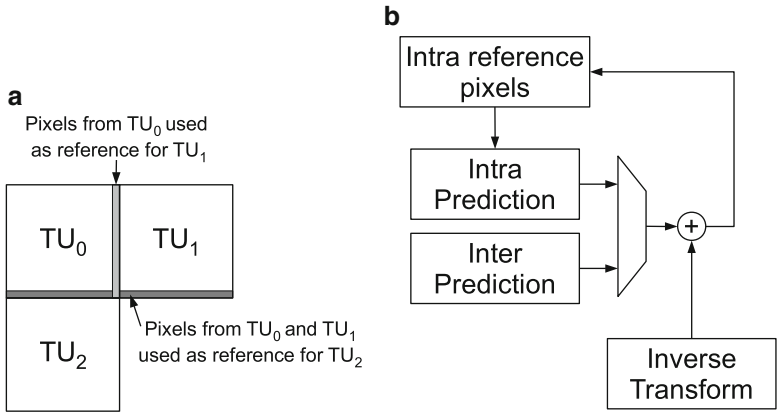


Fig. 10.19 Tight feedback loop in intra prediction due to dependency between neighbors. (**a**) Intra-prediction dependency between neighboring pixel blocks. (**b**) Dependency results in a tight feedback loop

The key operations in intra-prediction are:

1. Read neighboring pixels and perform padding for unavailable pixels
2. Reference preparation: filter neighboring pixels to obtain intra reference pixels and extend the top-left reference pixels for angular modes
3. Prediction: bilinear interpolation for angular and planar modes, and pixel copy for DC, horizontal and vertical modes

When the current block of pixels is predicted, its residues need to be immediately added so that it can be used as neighboring pixels for the next block. This results in a tight feedback loop for intra-prediction as shown in Fig. 10.19. As a result of this feedback loop, it is not possible to pipeline the above three operations, which increases the throughput requirement from these blocks. It should be noted that the feedback loop operates at a TU granularity and not a PU granularity. For example, for a 16 × 16 CU with a 2N × 2N intra partition (i.e. a single 16 × 16 PU) and a residue quad tree (RQT) of four 8×8 TUs, the 8×8 blocks must be predicted serially and the intra neighboring pixels must be updated after every block's prediction and reconstruction.

This dependency also has implications for the top-level pipelining—in order to keep inverse transform and prediction decoupled, the inverse transform must be performed one pipeline granularity before prediction.

The 35 intra prediction modes in HEVC are well designed to reduce complexity. The planar mode is much simpler than the one in H.264/AVC, and the 33 angular modes are also well organized to avoid increasing the complexity when increasing the angular precision. However, the larger TU sizes increase the hardware complexity due to larger pipeline and reference buffers. In H.264/AVC,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00340

one macroblock can contain only one kind of intra block size, which can be used to design optimized pipeline schedules as in [7,24]. Since a CTU in HEVC can have a variety of TUs and a mix of intra and inter CUs, such pipeline schedules will be too complex to optimize for every possible combination.

As the result, designing a data-flow that respects across-TU dependencies and provides high throughput is a bigger challenge than the pixel computation involved in reference preparation and prediction. In this chapter, we focus on the data-flow management used in [8], which uses a hierarchical memory deployment for high throughput and low area. The intra engine operates on blocks of 32×32 luma pixels and two 16×16 chroma pixels since those are the largest TU sizes. In the complete decoder pipeline, it communicates with entropy decoder and inverse transform at a Variable-sized Pipeline Block (VPB) granularity. (The mapping between VPB and CTU is shown in Table 10.1. For 16×16 CTU, four CTUs are combined into one intra pipeline block.)

10.7.1 Hierarchical Memory Deployment

The bottom row pixels of all VPBs in a row of VPBs needs to be stored since they are top neighbors for VPBs in the row below. This buffer must be sized proportional to the picture width and may be implemented in on-chip SRAM or external DRAM. Storing VPB-level neighboring pixels in registers as previous designs for H.264/AVC have done can provide the required high-throughput access. But this will require a lot of area as the VPB can be as large as 64×64. This issue can be addressed by storing the neighboring pixels in SRAM to save area and storing them in registers at a TU level for higher throughput. A memory hierarchy is thus formed:

1. VPB-row-level top neighbors in SRAM or external memory
2. VPB-level neighboring pixels in SRAM
3. TU-level reference pixels in registers

The hierarchical memory deployment is shown in Fig. 10.20 and the memory elements are explained next:

1. VPB-Row top neighbors: In [9], this buffer is implemented in an on-chip SRAM that is shared with deblocking filter. The deblocking filter stores four top rows of which, intra prediction uses one row.
2. VPB top neighbors: This buffer is implemented using a pair of SRAMs in a ping-pong fashion. One SRAM is used in the intra-prediction of the current VPB. It is updated every TU with neighboring pixels for the next TU. At the same time, the other SRAM updates the VPB-Row top SRAM with pixels from the previous VPB and loads top row pixels for the next VPB. The size of each SRAM is 192 pixels (64 Y top + 32 Y top-right + 64 UV top + 32 UV top-right).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00341

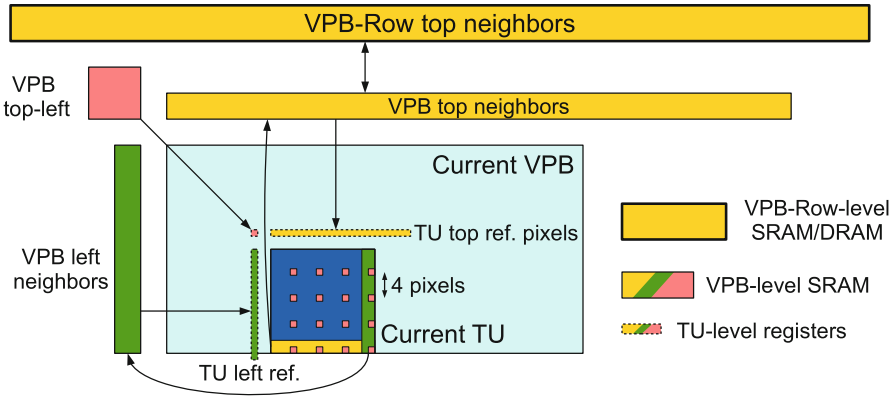


Fig. 10.20 Hierarchical memory deployment with VPB-Row level SRAM/DRAM and VPB-level SRAM for neighboring pixels, and TU-level registers for reference pixels

3. VPB left neighbors: This buffer is implemented using one SRAM containing 128 pixels (64 Y + 64 UV). It is updated every TU with neighboring pixels for the next TU. Because the TUs are processed in z-scan order, at the end of all TUs in the current VPB, it automatically contains the left neighbors for the next VPB.
4. VPB top-left neighbors: The TU-based update scheme for VPB top and left neighbors could overwrite some pixels which will be the top-left neighbor of some following TUs. The VPB top-left neighbor buffer is introduced to solve this problem. As shown in Fig. 10.20, pixels on the 4×4 grid are written to the VPB top-left neighbor buffer (Table 10.12).
5. Reference pixels: At the start of every TU, neighbors are read from the VPB-level SRAMs into registers. Padding and preparation operations are then performed on the registers to obtain reference pixels. Using registers allows for these operations and the final intra prediction to be performed at a high throughput. A total of 129 reference pixels (32 bottom-left, 32 left, one top-left, 32 top, 32 top-right) are needed for all angular modes. But since only one angular mode is used at a given time, the horizontal modes can be treated as vertical modes by swapping x and y axes to reduce the number of reference pixels to 99. Reference pixels are read by both preparation and prediction, and a combined read-out circuit shared between the two operations can reduce the number of multiplexers by exploiting similarities in their access patterns.

10.7.2 Reference Preparation and Prediction

As mentioned in Sect. 10.7, due to the tight dependency loop in Intra processing it is hard to pipeline the three pixel processing operations of reference padding,

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00342

Table 10.12 SRAMs for neighboring pixels

SRAM	Bits
VPB top	3072
VPB left	1024
VPB top-left	768
Total	4864

Table 10.13 Gate-count (in kgates) breakdown for Intra prediction

Module	Logic area
Reference pixel registers and padding	12.1
Reference pixel preparation	1.3
Prediction	8.1
Control	5.5
Total	27.0

reference preparation and prediction. Another factor is that the three operations require different amount of computation. For an $N \times N$ TU, reference padding and preparation require $O(N)$ computation while prediction is $O(N^2)$.

The reference preparation operation in HEVC varies depending on the prediction mode. DC mode requires the accumulation of the reference pixels in order to compute the DC value. An angular extension of the reference pixels may be required before prediction can begin. A mode dependant intra smoothing (MDIS) filter may be applied to the reference pixels for TU sizes 8, 16 and 32 depending on the intra mode.

10.7.3 Implementation Results

Table 10.13 shows the synthesis results for the intra prediction architecture in 40 nm CMOS. Reference pixel registers and their read-out take the most area. The area for reference preparation, which is a new feature in HEVC, is about 1.3 kgate. The design is synthesized at 200 MHz and can support 4K Ultra-HD decoding at 30 fps.

10.8 In-Loop Filters

HEVC uses two in-loop filters—deblocking filter and sample adaptive offset (SAO)—that attempt to reduce compression artifacts and improve coding efficiency. The deblocking filter in HEVC processes edges on an 8-pixel grid and thus, has lower computational complexity than H.264/AVC's deblocking filter which uses a 4-pixel grid. SAO involves selecting an offset type for each pixel based on its neighboring pixels and adding the offset. Deblocking and SAO can be implemented in a single pipeline stage as described in [30].

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00343

In [9], a VPB-based pipelining is used between deblocking filter and prediction stages. This allows the scheduling within the deblocking filter to be scheduled independent of the coding tree structure. A smaller granularity can also be used to save pipeline buffer SRAM at the cost of scheduling complexity. Since the in-loop filtering process for the current block of pixels depends on blocks to the right and bottom which have not yet been reconstructed, the entire block cannot be processed completely. The output of the deblocking filter is shifted from the input by four luma pixels and two chroma pixels to the left and the top, and the output of SAO is shifted by another pixel for all color components in both directions.

10.8.1 Deblocking Filter

Compared to H.264/AVC, HEVC's deblocking filter has several simplifications related to processing dependencies. The luma deblocking filter operates on edges lying on an 8×8 grid and filter takes 4 pixels on either side of the edge as input and writes up to 3 pixels on either side. As a result, unlike H.264/AVC, filters on adjacent edges are completely decoupled and it is possible to filter 8×8 pixel blocks independently. The key challenge in the deblocking filter architecture is designing an efficient data flow to handle cross-CTU dependencies.

The bottom four rows and right-most four columns of luma pixels (and two rows and columns of chroma pixels) in a CTU depend on the CTUs to the bottom, right and bottom-right for their deblocking. Accordingly, their processing must be delayed until those CTUs are available and they must be temporarily stored until then. Along with the pixels, parameters such as prediction mode, motion vectors, TU and PU boundaries, and quantization parameter which are required for computing the boundary strength also need temporary storage. The right-most four columns need a 1-CTU-high buffer (called Last CTU buffer) while the bottom four rows need a 1-Picture-wide buffer (called Line buffer).

The boundary strength parameters are available at a worst-case granularity of 4×4 pixels and take about 78 bits (64 bits for two motion vectors, 4 bits for two reference list indices, 6 bits for quantization parameter, 2 bits for prediction mode—intra-prediction, uni-prediction, bi-prediction—and one bit each for TU boundary, PU boundary). For example, for a 4K Ultra-HD ($3{,}840 \times 2{,}160$) picture and 64×64 CTU, the Last CTU buffer must hold 64×4 luma pixels, $2 \times 32 \times 2$ chroma pixels and 16 boundary strength parameters resulting in a total of 4,320 bits. The Line buffer must hold $3{,}840 \times 4$ luma pixels, $2 \times 1{,}920 \times 2$ chroma pixels and 960 boundary strength parameters resulting in a total of 96 kbit. While the Last CTU buffer can be stored in registers or SRAM, it might be necessary to store the Line buffer in external DRAM depending on area constraints. However, due to the regular access pattern on the Line buffer, it is possible to prefetch the data and hide the DRAM bandwidth (at the cost of on-chip memory for request and response queues to and from the DRAM).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00344

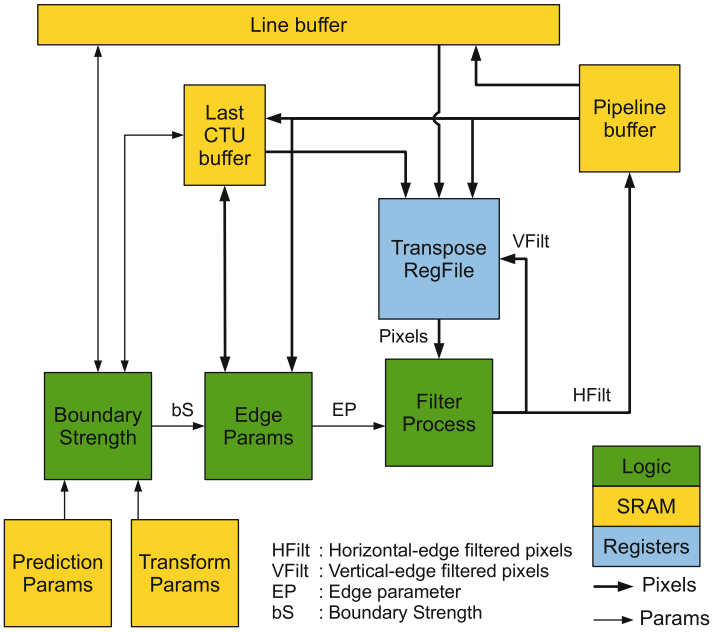


Fig. 10.21 Top-level architecture of deblocking filter

The top-level architecture of the deblocking filter is shown in Fig. 10.21. The transpose memory needs to be only 8 × 8 pixels (as compared to 32 × 32 pixels for inverse transform). Hence it is possible to implement it using registers. For a very high throughput design which filters an entire 8 × 8 block in one cycle [30], it is possible to eliminate the transpose memory completely and have a purely combinational design.

10.8.2 Sample Adaptive Offset (SAO)

SAO classifies each pixel into one of four bands or one of four edge types and adds an offset to it. For band offsets, the band of each pixel depends on its value and the position of the four bands. For edge offsets, the edge of each pixel depends on the whether its value is larger or smaller than two of its neighbors. The selection between band offsets and edge offsets, position of bands, choice of neighbors for edge offsets, and values of the offsets are signaled at the CTU level for luma and chroma separately. For chroma, the offsets are also signaled for the two components separately.

SAO has dependencies on neighboring pixels similar to intra prediction and hence, a similar data-flow management must be used. Like intra prediction, a picture-width sized top row buffer and a CTU-height sized left column buffer are

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00345

needed. These buffers store pre-SAO pixels and their SAO parameters. However, unlike intra prediction, the choice of pipeline granularity is very flexible and can be chosen based on throughput requirements. Unlike deblocking filter which operates on a edge basis, SAO operates on a per-pixel basis. So, the two in-loop filters have a comparable computational complexity even though SAO computation involves mainly comparison and addition.

Zhu et al. [30] describes an architecture for SAO that is capable of 8K Ultra-HD (7,680 × 4,320) at 120 fps. In spite of such high throughput requirement, the design takes only 36.7 kgates in 65 nm technology.

10.9 Implementation Results for Decoder Test Chip

A decoder test chip was implemented in [9] with a core size of 1.77 mm^2 in 40 nm CMOS, comprising 715K logic gates and 124 kB of on-chip SRAM. Figure 10.22 shows the micrograph of the test chip. It is compliant to HEVC Test Model (HM) 4.0, and the supported decoding tools in HEVC Working Draft (WD) 4 are listed in Table 10.14 along with the main specs. The main differences from the final version of HEVC are that SAO is absent and Context-Adaptive Variable

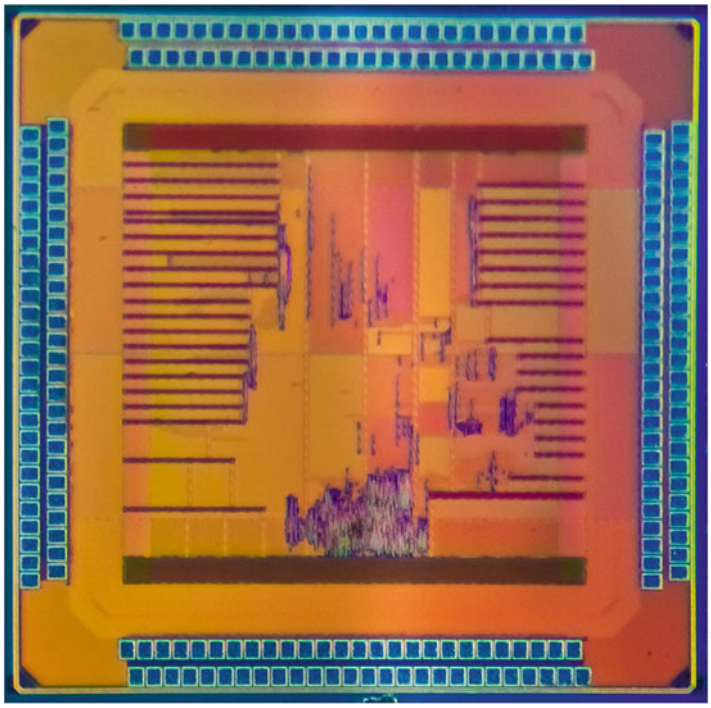


Fig. 10.22 Chip micrograph

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00346

Table 10.14 Chip specifications

Technology	TSMC 40 nm CMOS
Supply Voltage	Core: 0.9 V, I/O: 2.5 V
Chip Size	2.18mm × 2.18mm
Core Size	1.33mm × 1.33mm
Gate Count	715K (2-input NAND)
On-Chip SRAM	124 kB
Maximum Throughput	249 Mpixel/s @ 200 MHz
Decoding Tools	HEVC WD4 (HM 4.0 low complexity w/o SAO)
	CTU size: 64 × 64, 32 × 32, 16 × 16
	B-frame: Low Delay(LD)/Random Access(RA)
	Symmetric and asymmetric motion partitions: 4 × 4 — 64 × 64
	Square and non-square transform units: 4 × 4 — 32 × 32
	All intra modes: DC, Planar, 33 Angular, LMChroma
Measured Core Power	76 mW @ 0.9 V 200 MHz, 3840 × 2160 @ 30fps (average)
	51 mW @ 0.9 V 100 MHz, 1920 × 1080 @ 60fps (average)
	31 mW @ 0.9 V 25 MHz, 1280 × 720 @ 30fps (average)

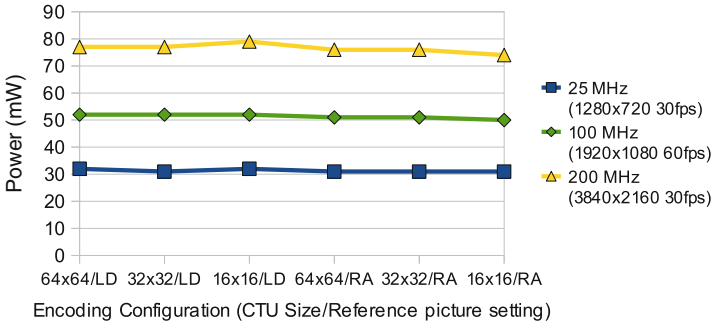


Fig. 10.23 Core power is measured for six different combinations—Random Access and Low Delay encoder configurations each with all three sizes of coding tree units. The core power is more or less constant due to our unified design

Length Coding (CAVLC) is used in place of CABAC in the Entropy Decoder. This chip achieves 249 Mpixels/s decoding throughput for 4K Ultra HD videos at 200 MHz with the target DDR3 SDRAM operating at 400 MHz. The core power is measured for six different configurations as shown in Fig. 10.23. The average core power consumption for 4K Ultra HD decoding at 30 fps is 76 mW at 0.9 V which corresponds to 0.31 nJ/pixel. Logic and SRAM breakdown of the chip is shown in Fig. 10.24. Similar to H.264/AVC decoders, we observe that prediction has the most significant resource utilization. However, we also observe that inverse transform is now significant due to the larger transform units while deblocking filter is relatively small due to simplifications in the standard. Power breakdown from post-layout power simulations with a bi-prediction bitstream is shown in Fig. 10.25. We observe

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00347

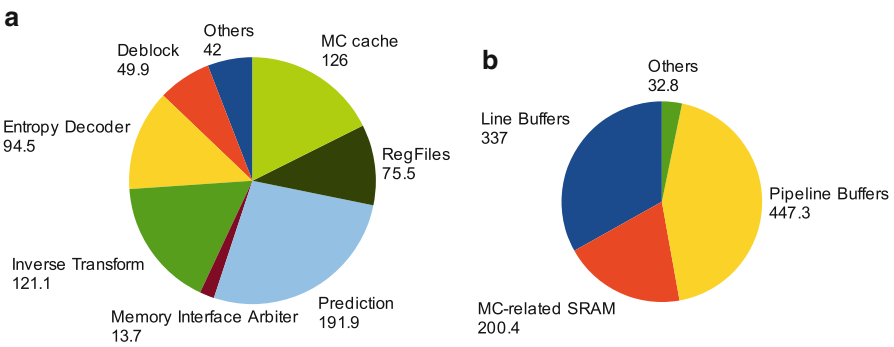


Fig. 10.24 Logic and SRAM utilization for each processing engine. (**a**) Logic utilization in kgates (total 715 kgate). (**b**) SRAM utilization in kbits (total 1,018 kbit)

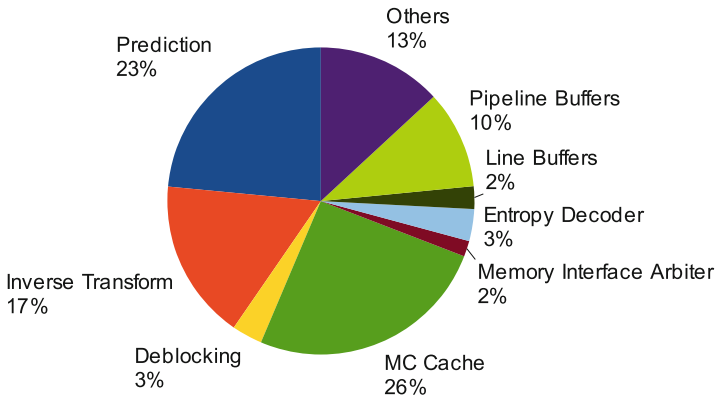


Fig. 10.25 Relative power consumption of processing engines and SRAMs from post-layout simulation with bi-prediction

that the MC cache takes up a significant portion of the total power. However, the DRAM power saving due to the cache is about six times the cache's own power consumption.

Table 10.15 shows the comparison with state-of-the-art video decoders. We observe that the 2× compression efficiency of HEVC comes at a proportionate cost in logic area. The SRAM utilization is much higher due to larger coding units and use of on-chip line-buffers.

10.10 Conclusion

This chapter presented the key challenges in implementing a hardware decoder for HEVC and techniques to address the challenges. The architecture of a test chip was described in detail. The test chip uses a variable-sized split system

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00348

Table 10.15 Comparison with state-of-the-art video decoders

	HEVC test chip [9]	A-SSCC'13 [20]	ISSCC'12 [29]	JSSC'11 [28]	ISSCC'10 [3]	JSSC'09 [19]	ISSCC'07 [2]
Standard	HEVC WD4	HEVC	H.264/AVC HP/MVC	H.264 HP	H.264/AVC HP SVC/MVC	H.264/AVC BP	JPEG, MPEG-1/2, MPEG-4, H.264 BP
Maximum Specification	3840 × 2160 @ 30 fps	1920 × 1080 @ 35 fps	7860 × 4320 @ 60 fps	4096 × 2160 @ 60 fps	4096 × 2160 @ 24 fps	1280 × 720 @ 30 fps	1920 × 1088 @ 30 fps
Gate count	715K	447K	1338K	662K	414K	315K	252K
On-chip SRAM	124KB	10KB	80KB	60KB	9KB	17KB	5KB
Technology	40 nm/0.9 V	90 nm/1.0 V	65 nm/1.2 V	90 nm/1.0 V	90 nm/1.0 V	65 nm/0.7 V, 0.85 V	130 nm/1.2 V
Core power	76 mW	139 mW	410 mW	189 mW	60 mW	1.8 mW	71 mW
Normalized core power	0.31 nJ/pixel	1.92 nJ/pixel	0.21 nJ/pixel	0.36 nJ/pixel	0.28 nJ/pixel	0.07 nJ/pixel	1.13 nJ/pixel
Normalized DRAM power	0.88 nJ/pixel	N/A	1.27 nJ/pixel	1.11 nJ/pixel	N/A	N/A	N/A
Normalized system power	1.19 nJ/pixel	N/A	1.48 nJ/pixel	1.47 nJ/pixel	N/A	N/A	N/A
DRAM configuration	32b DDR3	N/A	64b DDR2	64b DDR	N/A	ZBT SRAM	SDR

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00349

pipeline to process the wide range of Coding Tree Unit sizes and account for variable DRAM latency. The challenge of large and varied sizes of Transform Units can be addressed using Multiple Constant Multiplication and an SRAM-based transpose memory for an area-efficient implementation. Similarly, the use of Time-Multiplexed Multiple Constant Multiplication to optimize HEVC's longer interpolation filter was described. The longer interpolation filter also results in increased bandwidth requirement from reference picture buffer which is addressed by a cache and a DRAM-latency aware memory mapping. The design of a hierarchical memory organization was described to handle the pixel flow for intra-prediction and the main considerations for designing HEVC's in-loop filters were enumerated. Finally, simulated and measured power results for the test chip were shown.

Acknowledgements The authors gratefully acknowledge the support of Texas Instruments for sponsoring the HEVC decoder test chip project and Taiwan Semiconductor Manufacturing Company (TSMC) University Shuttle program for manufacturing the chip.

References

1. Chen J-W, Lin C-C, Guo J-I, Wang J-S (2006) Low complexity architecture design of H.264 predictive pixel compensator for HDTV application. In: IEEE international conference on acoustics, speech and signal processing, vol 3, pp 932–935
2. Chien CD, Lin CC, Shih YH, Chen HC, Huang CJ, Yu CY, Chen CL, Cheng CH, Guo JI (2007) A 252kgate/71mW multi-standard multi-channel video decoder for high definition video applications. In: IEEE international solid-state circuits conference (ISSCC). Digest of Technical Papers, pp 282–603
3. Chuang T-D, Tsung P-K, Lin P-C, Chang L-M, Ma T-C, Chen Y-H, Chen L-G (2010) A 59.5mW scalable/multi-view video decoder chip for Quad/3D full HDTV and video streaming applications. In: IEEE international solid-state circuits conference (ISSCC). Digest of Technical Papers, pp 330–331
4. Finchelstein DF (2009) Low-power techniques for video decoding. Thesis, Massachusetts Institute of Technology
5. Finchelstein DF, Sze V, Chandrakasan AP (2009) Multicore processing and efficient on-chip caching for H.264 and future video decoders. IEEE Trans Circuits Syst Video Technol 19(11):1704–1713
6. Guo Z, Zhou D, Goto S (2012) An optimized MC interpolation architecture for HEVC. In: IEEE international conference on acoustics, speech and signal processing, pp 1117–1120
7. He X, Zhou D, Zhou J, Goto S (2009) High Profile intra prediction architecture for H.264. In: IEEE international SoC design conference, pp 57–60
8. Huang C-T, Tikekar M, Chandrakasan AP (2014) Memory-hierarchical and mode-adaptive HEVC intra prediction architecture for quad full HD video decoding. IEEE Trans VLSI Syst
9. Huang C-T, Tikekar M, Juvekar C, Sze V, Chandrakasan A (2013) A 249Mpixel/s HEVC video-decoder chip for Quad Full HD applications. In: IEEE international solid-state circuits conference. Digest of Technical Papers, pp 162–163
10. Kawakami K, Takemura J, Kuroda M, Kawaguchi H, Yoshimoto M (2006) A 50 % power reduction in H.264/AVC HDTV video decoder LSI by dynamic voltage scaling in elastic pipeline. IEICE Trans Fundam Electron Commun Comput Sci E89-A(12):3642–3651

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00350

11. Lin CC, Guo JI, Chang HC, Yang YC, Chen JW, Tsai MC, Wang JS (2006) A 160kgate 4.5kB SRAM H.264 video decoder for HDTV applications. In: IEEE international solid-state circuits conference (ISSCC). Digest of Technical Papers, pp 1596–1605
12. Lin P-C, Chuang T-D, Chen L-G (2009) A branch selection multi-symbol high throughput CABAC decoder architecture for H.264/AVC. In: 2009 IEEE international symposium on circuits and systems, pp 365–368
13. Marpe D, Schwarz H, Wiegand T (2003) Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans Circuits Syst Video Technol 13(7):620–636
14. Micron. DDR3 SDRAM system-power calculator (2014) Available: http://www.micron.com/products/support/power-calc
15. Multiplexed Multiplier Block Generator (2014) Available: http://www.spiral.net/hardware/mmcm.html
16. Sze V (2010) Parallel algorithms and architectures for low power video decoding. PhD thesis, Massachusetts Institute of Technology
17. Sze V, Budagavi M (2012) High throughput CABAC entropy coding in HEVC. IEEE Trans Circuits Syst Video Technol 22(12):1778–1791
18. Sze V, Budagavi M (2013) A comparison of CABAC throughput for HEVC/H.265 VS. AVC/H.264. In: 2013 IEEE workshop on signal processing systems (SiPS), pp 165–170
19. Sze V, Finchelstein DF, Sinangil ME, Chandrakasan AP (2009) A 0.7-V 1.8-mW H.264/AVC 720p video decoder. IEEE J Solid-State Circuits 44(11):2943–2956
20. Tsai C-H, Wang H-T, Liu C-L, Li Y, Lee C-Y (2013) A 446.6k-gates 0.55–1.2v H.265/HEVC decoder for next generation video applications. In: 2013 IEEE Asian solid-state circuits conference (A-SSCC), pp 305–308
21. Tummeltshammer P, Hoe JC, Puschel M (2007) Time-multiplexed multiple-constant multiplication. IEEE Trans Comput Aided Des Integr Circuits Syst 26(9):1551–1563
22. Vanne J, Viitanen M, Hamalainen TD, Hallapuro A (2012) Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. IEEE Trans Circuits Syst Video Technol 22(12):1885–1898
23. Xanthopoulos T (1999) Low power data-dependent transform video and still image coding. Thesis, Massachusetts Institute of Technology
24. Xu K, Choy C-S (2008) A power-efficient and self-adaptive prediction engine for H.264/AVC decoding. IEEE Trans VLSI Syst 16(3):302–313
25. Yang YC, Guo JI (2009) High-throughput H.264/AVC high-profile CABAC decoder for HDTV applications. IEEE Trans Circuits Syst Video Technol 19(9):1395–1399
26. Yi Y, Park I-C (2007) High-speed H.264/AVC CABAC decoding. IEEE Trans Circuits Syst Video Technol 17(4):490–494
27. Zhang P, Xie D, Gao W (2009) Variable-bin-rate CABAC engine for H.264/AVC high definition real-time decoding. IEEE Trans VLSI Syst 17(3):417–426
28. Zhou D, Zhou J, He X, Zhu J, Kong J, Liu P, Goto S (2011) A 530Mpixels/s 4096x2160@60fps H.264/AVC High Profile video decoder chip. IEEE J Solid-State Circuits 46(4):777–788
29. Zhou D, Zhou J, Zhu J, Liu P, Goto S (2012) A 2Gpixel/s H.264/AVC HP/MVC video decoder chip for super hi-vision and 3DTV/FTV applications. In: IEEE international solid-state circuits conference (ISSCC). Digest of Technical Papers, pp 224–226
30. Zhu J, Zhou D, He G, Goto S (2013) A combined SAO and de-blocking filter architecture for HEVC video decoder. In: 20th IEEE international conference on image processing (ICIP), pp 1967–1971
31. Chen Y-H, Sze V (2014) A 2014 Mbin/s deeply pipelined CABAC decoder for HEVC. IEEE international conference on image processing (ICIP)
32. Tikekar M, Huang C-T, Sze V, Chandrakasan A (2014) Energy and area-efficient hardware implementation of HEVC inverse transform and dequantization. IEEE international conference on image processing (ICIP)
33. Huang C-T, Juvekar C, Tikekar M, Chandrakasan AP (2013) HEVC interpolation filter architecture for quad full HD decoding. In Visual Communications and Image Processing (VCIP), pp 1–5

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00351

Chapter 11
Encoder Hardware Architecture for HEVC

Sung-Fang Tsai, Cheng-Han Tsai, and Liang-Gee Chen

Abstract In this chapter, an encoder hardware architecture design for HEVC is described. The system pipeline is first introduced followed by the design details of the different HEVC encoder modules such as inter prediction, intra prediction, mode decision, in-loop filters, and entropy coding. Finally, a sample test chip implementation result is presented as a reference.

11.1 Introduction

High density large-sized displays that provide an immersive display experience are being widely adopted in multimedia application terminals. The next generation display panels are expected to be 4K/8K (Ultra High Definition Television (UHDTV) resolution) or even higher. As a result, the video resolution required is also becoming higher. To reduce the storage and transmission requirements of large-sized video, HEVC video encoding, that provides around 50 % better compression efficiency than H.264/AVC, is recommended. However, real-time high resolution video encoding in HEVC requires hardware support due to the complexity.

The HEVC Test Model (HM) is the reference software for HEVC developed during standardization of HEVC. The HM encoder was designed to maximize the coding gain achievable by HEVC. Real-time performance was not a major objective. The HM encoder can be used for a wide variety of applications including offline encoding [30]. However, the encoding methods in HM do not necessarily map to the most efficient design for a real-time hardware encoder. Hardware consideration is quite different from software and is very challenging. For H.264/AVC, many state-of-the-art H.264/AVC encoder architectures [4, 5, 18, 23, 43] have been presented.

S.-F. Tsai (✉) • C.-H. Tsai • L.-G. Chen
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
e-mail: sftsai@video.ee.ntu.edu.tw; phenom@video.ee.ntu.edu.tw; lgchen@cc.ee.ntu.edu.tw

V. Sze et al. (eds.), *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Integrated Circuits and Systems, DOI 10.1007/978-3-319-06895-4_1
© Springer International Publishing Switzerland 2014

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00352

However, the new features in HEVC have some impact on system architecture and have not been well addressed before. For example, HEVC increases the choice of both intra and inter prediction modes and the ranges of block sizes, which makes encoder mode decision much more complex than before. In HM, highly complex full rate-distortion optimization (RDO) is used to choose the final modes. Although high coding performance is achieved with full RDO, the hardware cost also becomes very high. For a practical real-time hardware encoder, these issues need to be addressed. Implementation costs should be in an acceptable range. In this chapter the coding losses from changes to the encoding algorithm to make it implementation friendly will be reported for low delay P conditions on 8K UHDTV video sequences based on HM 4.0.

11.2 System Pipeline

In HEVC, the basic processing block is the Coding Tree Unit (CTU). Similar to the macroblock pipeline in H.264/AVC encoders, a CTU processing pipeline can be applied to an HEVC encoder. The pipeline granularity is the largest CTU size, which can be up to 64×64.

Compared to H.264/AVC, HEVC has more complex combinations of prediction modes and prediction unit (PU) sizes. The HEVC encoder needs to perform a more precise mode decision, which uses bit rate estimates based the on entropy coding and distortion estimates based on the sum of squared distance (SSD). Due to the computation of the full RDO, a high complexity mode decision hardware (HCMD) that is separate from the prediction stages is needed.

HEVC also has a new in-loop filter called Sample Adaptive Offset (SAO), which adaptively adds offsets to reconstructed samples after deblocking filter. The encoder derives the offsets by minimizing the difference between the original input frame and the reconstructed frame after deblocking. The offsets and other parameters that control the SAO operation are then transmitted in the bitstream at a CTU level. Therefore, the encoding pipeline should be changed to ensure that SAO parameter derivation is finished before entropy coding.

Entropy coding should be parallelized to achieve the high throughput needed for high resolution video. HEVC provides several ways to parallelize entropy coding. The bitstream can be divided into multiple sub-streams, with limited dependency among the sub-streams. However, serial processing nature of entropy coding still prevails inside each of the sub-streams. Pipeline structure and the CTU processing order must be considered to perform parallel entropy coding in limited resource (e.g. memory size and bandwidth).

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00353

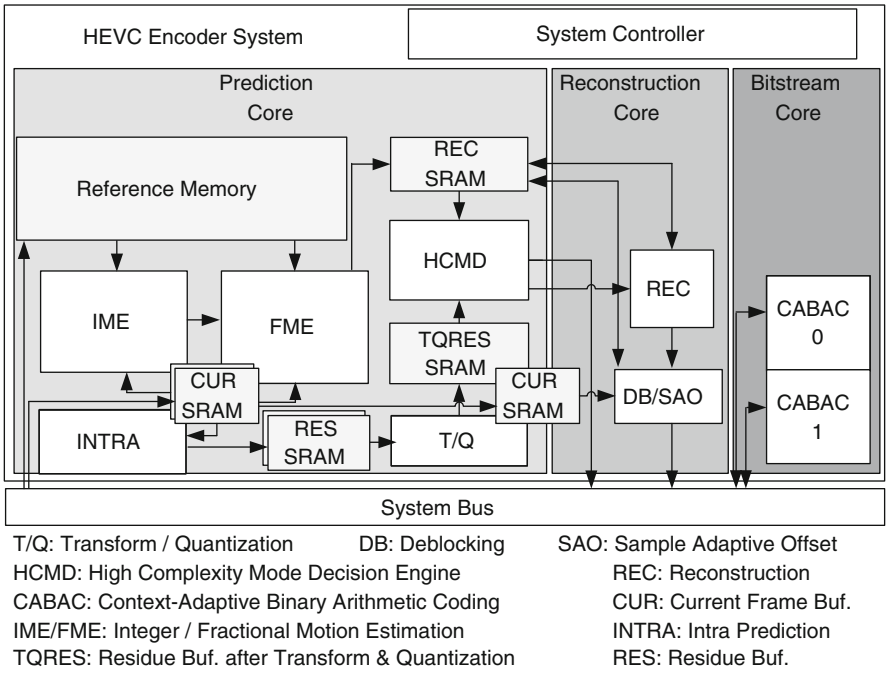


Fig. 11.1 Encoder system block diagram

11.2.1 Top Level System Diagram

An encoder system block diagram is shown in Fig. 11.1. It is composed of three major parts: the prediction core, the reconstruction core, and the bitstream core. Overall, the pipeline is longer than typical H.264/AVC pipelines to account for the processing dependencies and to increase heterogeneous parallelization between functional blocks.

The prediction core is the most important and computation-intensive part. It includes intra prediction, integer/fractional motion estimation (IME/FME) for inter prediction, transform and quantization, and mode decision hardware. In addition, a reference memory system is required to support the reference frames access of motion estimation. In this core, intra prediction and inter prediction are done in parallel. Dependency in between these two modules are removed as discussed in Sect. 11.4.2. CTU-sized current block buffer and residue block buffer are shared between stages with round-robin style buffer multiplexing. The motion estimation is performed in two levels of accuracy and is separated into two pipeline stages. In HEVC, the transform is more complex than in H.264/AVC and is put in a separate stage. If full RDO is used, a standalone HCMD stage will be included.

The reference frames are prepared by the reconstruction core every time a frame is encoded. The reconstruction core constructs the decoded reference frame as

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00354

seen by a decoder. It consists of inverse transform, inverse quantization, intra/inter prediction reconstruction, and loop filters (deblocking and SAO filters). Note that the reconstruction core may share the same hardware with the prediction core, or retrieve the results from the prediction core. It does not need significant additional cost as a standalone decoder.

Finally, the bitstream core performs entropy coding and writes out the final bitstream. In HEVC, the entropy coding is Context-Adaptive Binary Arithmetic Coding (CABAC). Note that SAO parameters should be encoded in the bitstream and so SAO parameter derivation should be done before CABAC encoding stage.

11.2.2 CTU Processing Order

The CTU processing order in the encoder pipeline will affect the reference data bandwidth and CABAC throughput. There are primarily two modules that are impacted by the CTU processing order: CABAC and reference memory subsystem. Change in CTU processing order will alter the data input order in CABAC. Due to the probability updates in the CABAC, the encoder and decoder must perform entropy coding on the CTUs in the same order. Thus the CTUs in the encoder can only be entropy coded in a defined order as determined by the HEVC specification (e.g. raster scan in slices or tiles). On the other hand, a change in the CTU processing order also alters the behavior of the reference memory subsystem. If the CTUs in the given processing order have better data locality, then the external bandwidth for reference frames access is lower.

11.2.2.1 Pipeline Granularity

In considering the precedence constraint of CABAC input data, there are two options for pipeline arrangement of CABAC. The first option is to put CABAC in the CTU pipeline. The CABAC engine must process the next CTU data generated by prediction engine strictly in order. The throughput requirement for CABAC with CTU-level pipelining is *peak* binary symbol (bin) rate per CTU. If such a throughput cannot be reached, the whole pipeline is stalled waiting for CABAC to complete thereby hurting the overall performance. In the second option, the CABAC is placed in a separate frame-level pipeline. Input data for CABAC is stored externally at first. After the whole frame passes through prediction and reconstruction, CABAC starts coding. This enables the prediction engine and the CABAC engine to process the CTUs in a different order within each frame. The throughput requirement for CABAC with frame-level pipelining is peak bin rate per frame, or equivalently *average* bin rate per CTU for the peak frame. This is generally much lower than the one with CTU-level pipelining. With this arrangement, the CTU processing orders in CABAC stage and in the other stages are independent, at the cost of extra external bandwidth.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00355

11.2.2.2 Parallel Processing

The HEVC standard requires that the context probability updates in the CABAC occur in the raster scan order. Raster scan order can be done with a single CABAC engine or multiple CABAC engines by using multiple slices/tiles within a given frame. A higher coding efficiency can be achieved with a single CABAC; however as only a single CTU can be processed at a time, and the throughput is limited. The CABAC needs to reach the *peak* bin rate per CTU, which can be quite high. Alternatively, multiple CTUs can be processed in parallel using slices/tiles. This comes at a cost of reduced coding efficiency since redundancy cannot be removed across slices/tiles. If only a few slices/tiles are used, then the coding loss can be quite low. Note that wavefront parallel processing, a new feature in HEVC, can also be used by the CABAC to encode multiple CTU lines in parallel, with a lower coding penalty than slices/tiles.

11.2.2.3 Data Locality

Another CTU scanning order that can be used in the encoder is the zigzag scanning order wherein the prediction core and the reconstruction core operate on the CTUs in the zigzag scanning order. However, the CABAC encoding will still need to happen in the raster scanning order in order to comply with the HEVC standard. For zigzag scanning order, the data locality among horizontal CTUs and vertical CTUs is better than raster scan which only has good data locality among horizontal CTUs. Zigzag scanning order cooperates well with CABAC frame-level pipelining discussed in Sect. 11.2.2.1. The difference is in the reference memory subsystem. Since data locality among vertical CTUs is better, zigzag scanning order performs better when the total on-chip memory size for reference frames is limited. Note that tiles, a new feature in HEVC, also offers better vertical and horizontal data locality by doing raster scan order within rectangular regions with widths smaller than the total frame width.

To summarize, the CTU processing order should be considered along with the various configurations of the parallel CABAC and the reference memory subsystem. The best choice is a trade-off among bit rate increase, area cost, throughput, and bandwidth.

11.3 Inter Prediction

In inter prediction, the temporal redundancy is reduced through motion estimation. Motion estimation compares the current prediction unit (PU) with the spatially neighboring PUs in the reference frames, and chooses the one with the least difference to the current PU. The displacement between the current PU and the matching PU in the reference frames is signaled using a motion vector. The per-pixel

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00356

difference between the current PU and the matching PU in the reference frames constitutes the prediction error (a.k.a. residue) which is transformed and quantized and coded in the bitstream.

In comparison with H.264/AVC, inter prediction in HEVC has three major differences: (1) larger diversity in block size, (2) high complexity mode decision is needed to achieve sufficient coding gain, and (3) longer sub-pixel interpolation filter. In HEVC, the PU size may range from $4 \times 8/8 \times 4$ to 64×64. Computation complexity for deciding the best block partition also increases considerably. To accurately choose the best mode among such high number of possible modes, full RDO invoking more accurate distortion and bit estimation needs to be applied. This requires inter predictions to preserve several possible modes for later HCMD stage. HEVC utilizes 8 or 7-tap interpolation filter for higher interpolation accuracy compared with 6-tap in H.264/AVC. So the complexity in sub-pixel calculation is also higher. To cope with these complexity increases, higher parallelism in hardware is necessary. This should be achieved with moderate cost increase. In addition, the parallelism in hardware also induces much higher memory access bandwidth. A memory subsystem that supports high bandwidth requirement is required to make motion estimation work properly. These issues are covered later in this section.

11.3.1 Motion Estimation

Due to the difference in the processing nature, inter prediction is usually divided into two major modules, integer motion estimation (IME) and fractional motion estimation (FME), corresponding to two granularity levels, the integer level and the fractional level. IME usually performs a coarse search over the whole search region. In this level, the parallelism requirement is high, while the accuracy requirement is moderate. After that, FME does a fine search around the IME searched result in sub-pixel accuracy. 8 or 7-tap interpolation filtering is required to get the pixels in the fractional positions. Since the distortion costs among neighboring sub-pixel candidates are similar, higher accuracy in the distortion computation is required in order to select the best candidate. The reference architecture is shown in Fig. 11.2.

In previous works, various architectures for variable block size motion estimation have been compared [7, 24]. A fast gradient-based algorithm on a parallel 2D SAD tree with high data reuse is described in [10]. Exploration in data reuse for motion estimation is shown in [13]. To increase parallelism, a highly parallel inter mode decision in HEVC is achieved by dependency removal in [41]. Finally, [35] describes how throughput requirements can be met by processing multiple CUs in parallel, but processing the PU within each CU serially to achieve the same sequential order as in HM. The result shows small block sizes (e.g. 4×4, 4×8, 8×4) impose significantly larger hardware, but provide only modest improvements in coding efficiency. In addition, a search range strategy centered on the advanced motion vector predictors (AMVP) with pre-fetch and limited search range movement is presented.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00357

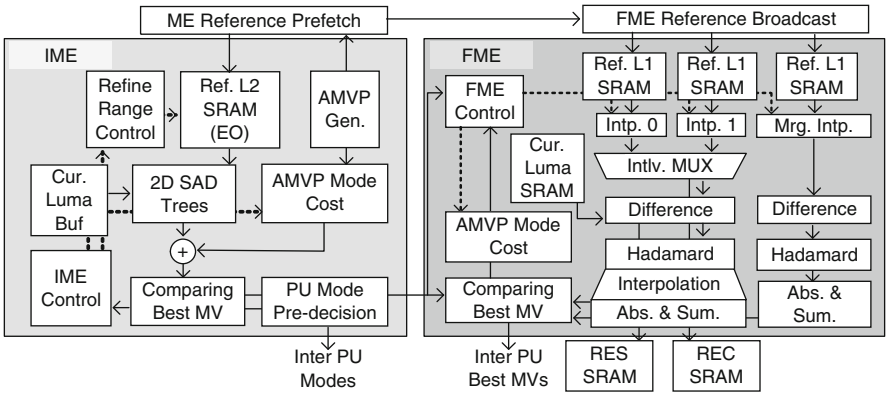


Fig. 11.2 HEVC inter prediction architecture

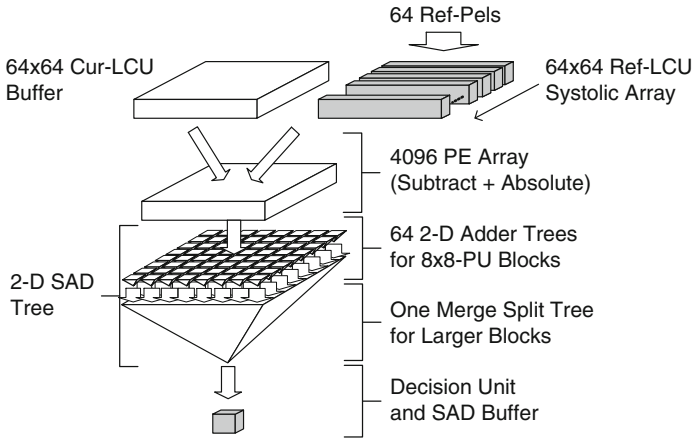


Fig. 11.3 2D SAD adder tree architecture

In this work, the IME architecture uses a parallel-PU IME architecture based on 2D SAD adder tree to meet the high throughput requirement [10], as illustrated in Fig. 11.3. In parallel-PU IME, all of the PU blocks inside current CTU are done in parallel. Instead of looping over all the pixels inside the PU block, the cost for bigger PU can be simply derived from the cost of the sub-divided PU (i.e. a bottom up approach). For example, retrieving the SAD cost of 16×8 PU can be simply done by adding the two co-located 8×8 PUs. By utilizing the 2D SAD adder tree, we may retrieve all PU costs for certain motion vectors at once. However, there are dependencies among PUs, such as reference motion vectors from the neighboring PUs used to calculate the predictor motion vector to use in motion vector cost calculation. To enable parallelism, we estimate the motion vector predictors from the nearest available vectors just outside of the current CTU. For the IME search algorithm, we need to select one of the two advanced motion vectors predictors

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00358

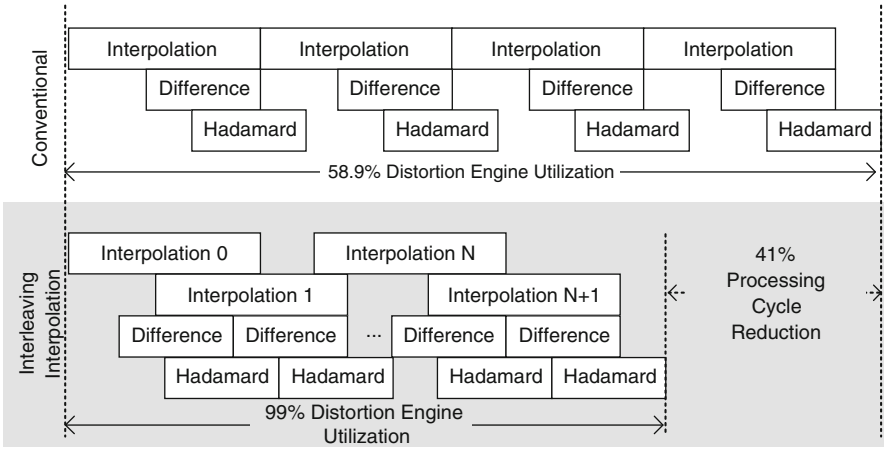


Fig. 11.4 The schedule of interleaving interpolation in FME

(AMVP). In hardware, we may use coarse-fine search around both the AMVPs. The search is performed within a small range, say $[-16, 15] \times [-16, 15]$, around both the AMVPs. First, candidates in the search area with even row and even column start address are searched. After that, a $[-1, 1] \times [-1, 1]$ refinement search will be performed around the best candidates. The motion vector predictor which leads to the lowest RD-cost is selected and signaled as the AMVP. In addition, pixel value bitwidth reduction (pixel truncation) can be used for reducing the cost. Five most significant bits for each pixel are used for estimating IME cost instead. Also, $2 \times 2 - to - 1$ (quarter) pixel sub-sampling can be done to further reduce the hardware cost. For every 2×2 pixels in the candidates, only upper-left pixel is used for estimating IME cost.

For FME, 8 or 7-tap filtering is required in fractional pixel interpolation. Compared with 6-tap filtering in H.264/AVC, the interpolation latency in HEVC is longer and causes low utilization for the other engines in FME. The difference and Hadamard engines are stalled for over 40 % of the cycles as shown in Fig. 11.4. To balance the pipeline throughput in FME, we apply an interleaving interpolation scheme. As Fig. 11.4 shows, the interpolation of the second block is started while the first block is being processed. Interpolation is carefully scheduled. Once the first block is finished, the first interpolated sample in the second block is ready. With this scheme, the number of stalls are significantly reduced. For FME, a two-pass fully utilized and reusable fractional motion estimation is presented in [6]. To improve throughput of the FME hardware, a one-pass central-quarter FME [9] can be applied for fractional-pixel estimation. Instead of the original sequential half-then-quarter refinement algorithm in HM, the one-pass central-quarter FME algorithm searches 25 half-pixel/quarter-pixel candidates around the best integer-pixel candidate at the same time to facilitate parallel processing and data reuse as indicated in Fig. 11.5. Twenty five processing units process 25 half-pixel/quarter-pixel candidates in

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00359

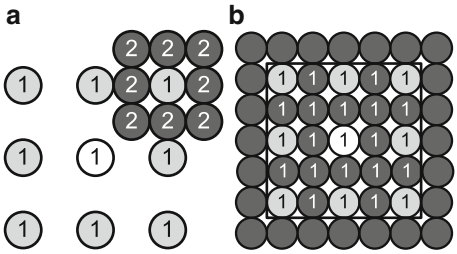


Fig. 11.5 Illustration of central-quarter fractional motion estimation algorithm. The *white, light-gray, dark-gray circles* are the best integer-pixel candidates, half-pixel candidates, and quarter-pixel candidates, respectively. The numbers represent the passes. (**a**) Two-step FME; (**b**) Central-quarter one-pass FME. The 25 candidates insides the *dark square* are processed in parallel

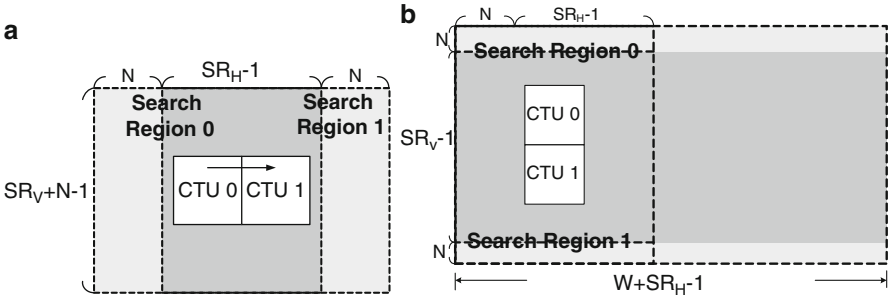


Fig. 11.6 The schemes of searching range data reuse (**a**) Level C (**b**) Level D

parallel. The interpolation can even be simplified to bilinear interpolation [39] for further cost reduction. The motion compensation is done after FME is finished. A total of 6.084 % BD-rate increase or equivalently 0.194 dB BD-PSNR degradation is observed for the described changes to IME and FME.

11.3.2 Reference Memory Subsystem

The major design challenge in supporting inter frame encoding in HEVC is the large bandwidth requirement for reference frame access. This issue will be discussed from both system-level and module-level views. From a system-level view, motion estimation in high resolution causes excess external bandwidth requirement. The huge memory bandwidth comes from loading the data of candidate blocks. To reduce the memory access, data reuse schemes such as Level C, Level D, and Level C+ are proposed [8, 22, 38]. A cache-based scheme to reduce the required SRAM size with similar external bandwidth as above is proposed in [40], at the cost of variable latency. For Level C scheme, the data is reused among search range of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00360

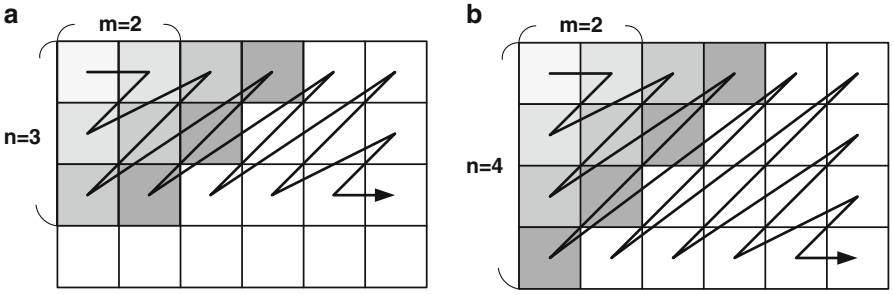


Fig. 11.7 Level C+ searching range data reuse (**a**) HF2V3 scan (**b**) HF2V4 scan

horizontal neighboring CTUs. As indicated in Fig. 11.6a, the light gray region is the search range of two neighboring CTU. The dark gray region is the overlapped search range. There is a large overlap region among search ranges of horizontal neighboring CTUs. The data in the common region is saved on chip in Level C scheme and thus redundant off-chip bandwidth is saved. The required SRAM size is the search range size. Level D scheme drives the data reuse even further. Since there is also overlap among vertical CTUs as indicated in Fig. 11.6b, more redundant access can be pruned if the vertical overlap data is saved on chip in this case. Thus, data reuse is driven from CTU level to CTU strip level in the Level D scheme. However, we need to store multiple CTU-strips owing to frame-level raster scan of CTU coding order. As a result, the SRAM size should include several rows of CTU strips, and so SRAM size would be much larger for Level D. Level C+ scheme aims to provide a continuous trade-off between Level C and Level D schemes. Level C+ scheme is based on the Level C scheme with larger SRAM size. The stored range in Level C+ scheme is several CTUs more than that in Level C scheme in both width and height. A zigzag scan of CTUs can be done in this buffer. Figure 11.7 shows the possible zigzag scan order without breaking dependency among the CTUs. Since coding a CTU requires information from the coding results of the upper row CTUs, several CTUs in the first row (i.e. m CTUs) must be coded first before the first CTU in the second row is coded. Thus, in the first step we perform a horizontal traversal for m CTUs. In the second step, the $(m+1)th$ CTU in the first row and the first CTU in the second row are ready and are traversed. In the third step, the $(m+2)th$ CTU in the first row, and the second CTU in the second row are ready and are traversed, and so on. To limit the traversed range within a limited vertical range in one strip, at most n rows are traversed. For rows outside the range, they will be covered by next traversal strip. With the zigzag scan, vertical data reuse can be performed in a limited range. The zigzag scan size is parameterized by m and n. m and n are decided by the data dependency among CTU and the data processing latency of CTU pipeline. If n is more than 1, we need to support zigzag scan order. In this case, the Level C+ SRAM is required to store additional $(m-1)$ CTU columns and $(n-1)$ CTU rows compared to the Level C scheme. By adjusting n, we may optimize the design trade-off between the SRAM size and the external bandwidth. With larger n, more

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00361

Table 11.1 The comparison of different data reuse schemes for ME [8]

Reuse scheme	EMB (Pixels/Pixel)	On-chip memory size (Pixels)
Level C	$1 + SR_V/N$	$(SR_H + N - 1) \times (SR_V + N - 1)$
Level C+	$1 + SR_V/nN$	$(SR_H + mN - 1) \times (SR_V + nN - 1)$
Level D	1	$(SR_H + W - 1) \times (SR_V - 1)$

EMB: External Memory Bandwidth of reference frame
SR_H: horizontal search range SR_V: vertical search range
N: current CTU size n: zigzag stitch number W: frame width

redundant access is saved, while the SRAM buffer will also be larger. By varying n, the Level C+ scheme provides a continuous SRAM and bandwidth trade-off and can adapt to the design requirements. In Table 11.1, the comparison is listed for the Level C, the Level C+ , and the Level D schemes. Note that cache-based scheme may also be applied to raster scan order or zigzag scan order in Level C+ to save more SRAM.

To illustrate a practical case, we take 8K UHDTV as an example. Assume motion estimation supports [−128,+127] search range and totally two reference frames. If we use the level C data reuse strategy, the bandwidth requirement will be as high as 11.61 GB/s for reference memory access. The level C strategy will discard the reference pixels that are out of the search range for the current CTU, and reload them later on while processing CTUs in the next CTU row, and thus will still consume high bandwidth in ultra high resolution sequences. The level C+ scheme or the level D scheme use reduced bandwidth. For the level D scheme, the bandwidth can be reduced to 2.97 GB/s.

From a module-level view, inter prediction in HEVC is very complex and requires high degree of parallelism. This also imposes significant internal on-chip memory bandwidth and multi-port access requirement. If high complexity mode decision is used, there will be multiple refinement levels of CU depth that need to be searched in fractional motion estimation (FME) stage. For each CU depth, merge candidates need to be searched and additional memory ports are required. In addition, if interleaving interpolation is used in FME stage, 2× the number of ports is required per refinement level. As a result, the required number of ports will be much higher than that in previous H.264/AVC encoders. For instance, a total 13 ports is required if IME and three-level refinement of FME are operating in parallel.

Considering the system-level and module-level view, a level C+ or a level D search window memory with high number of ports is required. However, highly-ported and large-sized memory is costly. SRAM banking is an alternatively used technique for increasing access parallelism. The case for SRAM banking is shown in Fig. 11.8. Each column is put on a separate SRAM bank. Each row corresponds to a separate SRAM address. For example, $A1$ is put on address #1 at bank #A, and $C4$ is put on address #4 in SRAM bank #C. With banking, we may access any combination that does not have bank conflict. For example, $\{B3,C4,D5,E6\}$ can be read out in one cycle without conflict, while $\{D5,E5,D6,E6\}$ cannot since $\{D5,D6\}$ uses the same bank D, and $\{E5,E6\}$ uses the same bank E. Each bank may serve only one read address per cycle. Thus, the read operation must

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00362

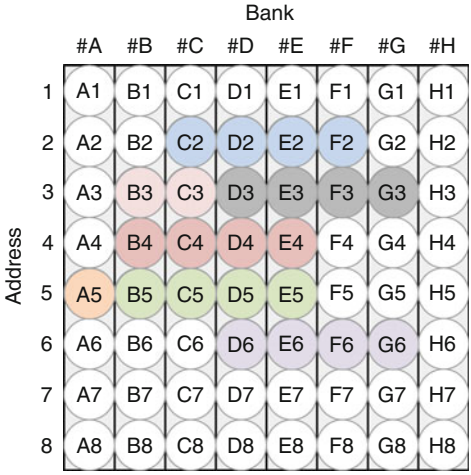


Fig. 11.8 Access conflict for reference memory banking

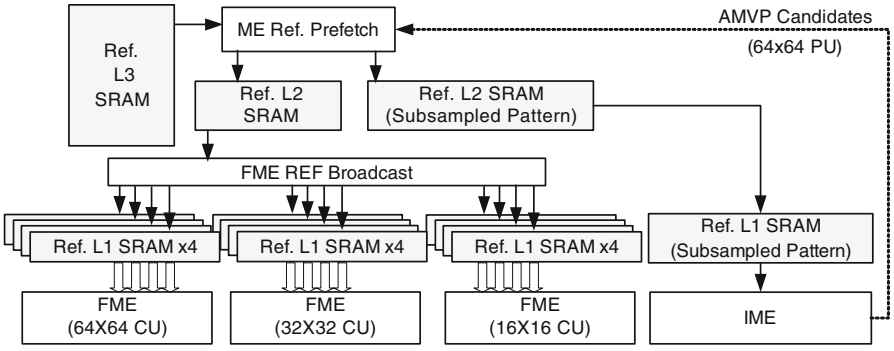


Fig. 11.9 Reference memory hierarchy subsystem architecture

be divided into two parts—$\{D5,E5\}$ and $\{D6,E6\}$. As a result, SRAM output bandwidth is lowered and causes IME/FME engine to be stalled until all the data is retrieved, causing performance loss. If this does not happen frequently, it is acceptable. However, the motion vectors are dependent on the input sequence, and bank conflicts may happen a lot. For example, IME/FME may need to read out $\{C2,D2,E2,F2\}$, $\{B3,C3,D3,E3\}$, $\{D3,E3,F3,G3\}$, $\{B4,C4,D4,E4\}$, $\{A5,B5,C5,D5\}$, $\{B5,C5,D5,E5\}$, $\{D6,E6,F6,G6\}$ at the same time due to concurrent operation of multiple parallel engines. Serious bank conflicts will occur in this case. To make matters worse, the conflict pattern may differ at different times according to the distribution of motion vectors. It is hard to find a specific pattern with good performance. Thus, SRAM banking is still not helpful to this issue. Therefore, a new reference memory strategy that may provide multi-port access and level D like data reuse is to be designed. The multi-level reference memory hierarchy subsystem is presented in Fig. 11.9 to fulfill the requirement of external

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00363

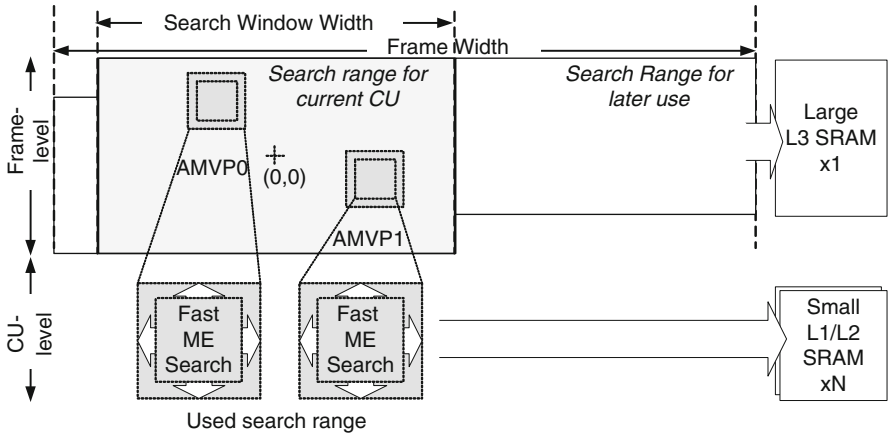


Fig. 11.10 Data granularity in reference frame access

bandwidth and internal bandwidth. We can see the data characteristics at various levels of data granularity as illustrated in Fig. 11.10. For a given CU, IME and FME that use fast algorithms may not access the whole search window memory. Instead, only small portions of the search range are accessed. At the module level, IME and FME do not require data outside the real search region. Larger memory results in larger area, higher power, and higher area, hence it is not efficient to store the whole search range for IME and FME use. For this reason, we use a multi-level and multiple reference memory with each level optimally resized for the best efficiency. A large L3 reference SRAM is used to enable level C+ /level D style buffering for lowest bandwidth overhead. For every pixel, the reference memory access reaches one read per frame if deep level C+ or level D is used. To support high concurrent access on the memory ports at the module level, we use L2 and L1 SRAM. The ME reference prefetch unit would fill the L2 SRAM for IME usage, and the L2 buffer for FME reference broadcasting unit. For IME that uses subsampling, the SRAM must be stored in subsampled pattern. The SRAM for IME is filled according to subsampling order for storing reference pixels. The FME reference broadcasting unit fills fully sampled L1 SRAM with data from the L2 buffer. With this scheme, all the concurrent access requirements from IME and FME are supported. The memory bandwidth is also minimized.

In addition, the architecture can be scaled up if more read ports are required. The total SRAM size needed for increasing the number of read ports is shown in Fig. 11.11. As an example, assume one set of IME engines and four sets of FME engines are used to meet certain design requirements. In this case, the reference memory hierarchy needs to support a total of 17 ports. We may achieve this simply by using 16 L1 SRAM with fully sampled pattern and 1 L1 SRAM with subsampled pattern. As shown in Fig. 11.11, if the number of ports increases by 30 % (i.e. from 13 to 17 ports), the additional reference memory size for supporting four more ports is only 2.7 % (from 7.14 to 7.33 MB). Thus, this architecture has high read port scalability.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00364

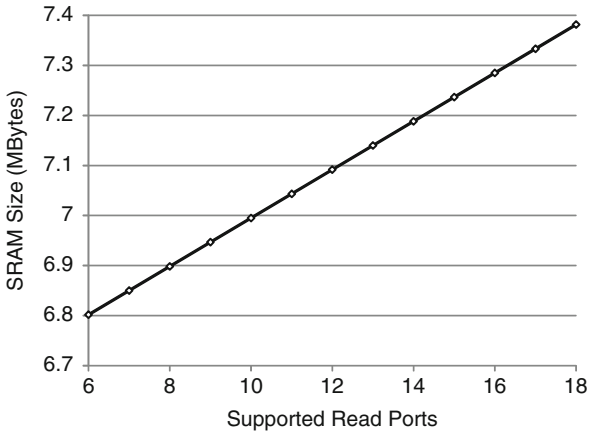


Fig. 11.11 Scalability for reference memory hierarchy. The SRAM size is the sum of all the L3, L2, and L1 reference SRAM size in the memory hierarchy. The ports are the supported concurrent read ports that can provide data to the processing engines. More read ports are supported by increasing the number of L1 SRAMs

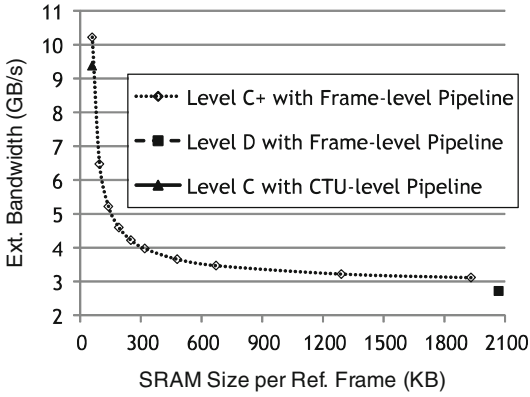


Fig. 11.12 Analysis of the strategy for search window data reuse

For the top L3 level memory, we did an analysis on the strategy for level C, level C+, and level D. The analysis result is shown in Fig. 11.12. Note that the level C+ strategy uses a non-raster scan CU order and requires frame-level pipeline at entropy stage. As can be seen in Fig. 11.12, the bandwidth of level C+ strategy is bigger initially, and drops down quickly below that of Level C strategy as the SRAM size increases. That is because frame-level pipelining is a must for level C+ strategy, so frame-level I/O overhead dominates at the beginning. However, the bandwidth gain for level C+ strategy quickly overrides the penalty from frame-level entropy pipelining. Depending on the requirement, level C+ or level D strategy is more suitable for L3 level memory.

The detailed data flow is shown in Fig. 11.13. IME requires the data in subsampled pattern, while FME requires data in fully sampled pattern. To support the two kinds of data pattern, we have two kinds of data ordering in SRAM. One is able to support subsampled pattern. The other is able to support fully sampled pattern.

Data exchange among stages is done with round-robin style memory multiplexing. Since the Ref. L2 SRAMs and the Ref. L1 SRAMs are of the same size, we put one set of L1 SRAM and L2 SRAMs in the multiplexing to save SRAM usage.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00365

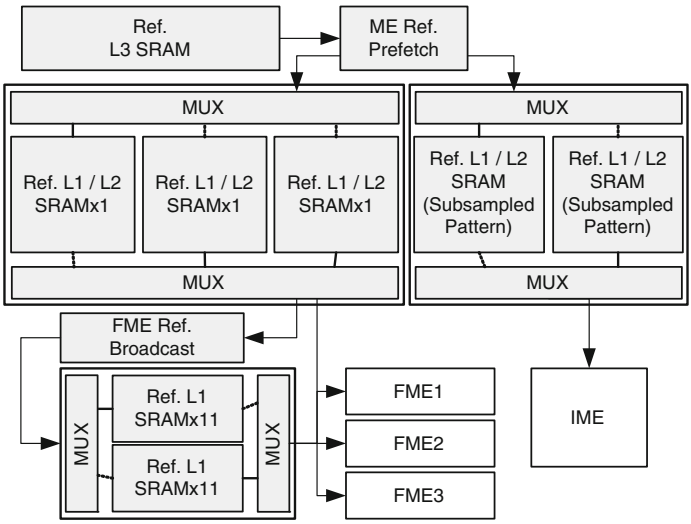


Fig. 11.13 Detail data flow for reference memory subsystem

Data scheduling for each stage is explained in the following. Before start, reference frame data within the maximum supported search range are loaded into L3 SRAM through system bus. In stage 1, the ME prefetching unit reads the actually used data from the L3 SRAM and writes them to the L2 SRAMs. There are two kinds of L2 SRAMs with different data ordering. The ME prefetching unit prepares the data in both kinds of order and writes them to the corresponding L2 SRAMs. After that, the connections to the L2 SRAM and the L1 SRAM are exchanged by altering the multiplexer configuration. The L2 SRAM in stage 1 will act as the L1 SRAM and be connected to the IME engine in stage 2. Thus, the written data in stage 1 can be accessed by the IME engine in stage 2. In stage 2, the IME engine reads out the data in the L1 SRAM with subsampled pattern. The FME reference broadcasting unit reads from the first L1 SRAM in fully sampled pattern and broadcasts the data to the other 11 L1 SRAMs. In stage 3, all the 12 L1 SRAMs are filled with reference frame data and are able to support three sets of FME engines with full sampled access.

11.4 Intra Prediction

Intra prediction reduces spatial redundancy in video by predicting from neighboring pixels around the current PU. The neighboring pixels used for prediction are the previously reconstructed pixels. Compared to H.264/AVC, there are primarily two differences: (1) Increased prediction modes and (2) Hybrid intra/inter prediction modes inside the same CTU. For the first one, there are a total of 35 possible intra

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00366

Table 11.2 Fast intra prediction results. All intra configuration is used

# of candidates	10	12	14	17
ΔBD-Rate[%]	0.21	0.19	0.06	0.03

modes for each PU size. The PU size may range from 4×4 to 64×64 depending on encoder configuration (for the 64×64 PU, the intra prediction actually happens at a 32×32 level since the maximum transform size supported is 32×32). The complexity is much higher and results in considerable hardware costs. In addition, HEVC allows the use of intra/inter CU, and consequently PU, inside the same CTU. The prediction dependency among PU is quite complex and limits the hardware parallelism. For hardware implementation of intra prediction, an architecture for intra 4×4 prediction with flexible reference sample selection is proposed in [28]. A full intra architecture is also proposed in [33]. In high throughput applications, intra prediction performance still needs to be improved. Here we apply two methods. Firstly, we use reduced mode search to lower the required computation at algorithm level. Next, hybrid open-closed loop intra prediction removes the dependency of reconstructed pixels and enables PU-level parallelism.

11.4.1 Reduced Mode Search

Depending on the configuration, intra PU size in HEVC may range from 4×4 to 64×64. For each size, the prediction directions can be chosen from 35 modes at most. Due to high number of modes, the search cost is high. To reduce the cost, we propose fast intra prediction algorithm, which will reduce two thirds of the total estimated modes per PU size. Only a limited number of modes are searched with constrained resource and timing budget.

Firstly, we assume the computation budget to be C. We are to find the best predicted modes within the computation budget of C modes. If the mode search cost reaches the limit, the search stops. The best result is taken as the predicted mode. To perform the search with efficiency, we use a two-step fast search algorithm. In the first step, we perform a coarse search. We search the whole range of directions but with an angular step size of $\pi/8$. The cost for each mode is obtained. In the second step, refinement search is performed at the unsearched angular neighboring modes that are just around the best one among the searched ones in the first step. The best mode is updated after each search. After that, an exhaustive search will be performed for every remaining mode till the budget limit is reached. Analysis of the trade-off between the quality and the number of searched modes is shown in Table 11.2. By using this algorithm, the number of searched modes is reduced to one third of the full search with negligible BD-rate increase.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00367

Modes in 64 × 64-sized PU are skipped in fast intra prediction. The main advantages from large PU size are in inter mode. When the frame size grows larger, the region in the same object and with the uniform movement can be predicted well and encoded economically with large block size. However, large-sized PU is less useful in intra. Practically, 64 × 64-sized PU is rarely chosen (less than 1 %). Therefore, the 64 × 64-sized intra prediction is not important and can be removed. With the removal of 64 × 64-sized intra prediction, the additional set of DCT and SRAM for 64 × 64-sized intra prediction is saved.

11.4.2 Hybrid Open/Closed Loop Intra Prediction

The reference pixels in intra prediction are obtained from the previously reconstructed neighboring PUs. In the HM software, the selected modes and CTU partitions are decided serially by full RDO process. However, full RDO process in hardware results inevitably in high latency because of the serial processing nature of CABAC. This will become the primary limiting factor of hardware design. It is hard to process all PUs serially as in HM and achieve enough processing throughput as well. In this work, we choose to remove the dependency to avoid these problems. Hybrid open-closed loop intra prediction algorithm [18] is useful in this case. Based on the fact that original pixels are close to reconstructed ones at low QP, this method uses the original pixels to replace the reconstructed boundary ones. In other words, if the reconstructed pixels are not yet ready, the original ones are used instead. In addition, intra/inter dependency in intra prediction is also removed in this case. Thus, the dependency does not exist in this algorithm. By hybrid open-closed loop scheme, intra mode for each block can be calculated in parallel without waiting for reconstruction. With all the modifications on intra prediction, the cost is 1.03 % BD-Rate increase in low delay P configuration.

11.5 Transform and Quantization

After the intra and inter prediction with early mode decision, the transform is performed. The corresponding block of residual samples is obtained from the difference between the original input samples and the predicted samples. It is then further processed by DCT-based coding with variable transform sizes. For 4 × 4 luma intra-prediction residual, the DST is used instead of the DCT. After transform, the residues are quantized and sent to entropy coding. Since the transform is rather complex in HEVC, quantization is done in a different stage. For transform, the size can be 4 × 4, 8 × 8, 16 × 16, and 32 × 32. Each transform unit (TU) can use the large transform size or be further subdivided into smaller transform sizes by a residual quadtree structure. The main reason for supporting different transform sizes is to adapt the transform to the varying space frequency characteristics of the residual

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00368

signal. To save on the transform decision cost, one of the options in HM is to use the largest available transform size. Some previous works [15] propose fast transform decision by early split termination. Owing to the fact that 32×32 TU is less selected than 8×8 TU and 16×16 TU, Teng [37] proposes a split-and-merge process. While this method works for software, it is not suitable for hardware implementation with variable hardware resource requirements and non-predictable computation cycles. For hardware architecture, a reusable architecture for various TU size is proposed in [31]. A unified engine for forward and inverse transform architecture for HEVC is shown in [2,45]. The distribution of transform residuals has strong relationship with the homogeneity of the predicted results. As a result, we choose to use the largest possible transform size inside PU boundary. With this fixed decision, the hardware cost for transform is reduced at the cost of BD-rate increase of 3.02 % in low delay P configuration.

11.6 Rate Distortion-Optimization

Rate distortion optimization (RDO) is an important method in video coding. It is based on the Lagrange optimization technique. With the proper choice of parameters, optimal trade-off between rate and distortion can be achieved. In HEVC, the combinations of prediction modes are much more complex than that in H.264/AVC. Low precision fast RDO is used in previous H.264/AVC encoders based on rough estimation of bit rate and distortion. The decision quality, however, is quite low. A more precise mode decision is thus required to maximize the coding gain. In HM, a full RDO method based on CABAC bit rate estimation and SSD distortion cost is used. However, the cost is very high in real-time video encoder hardware. In this section, we discuss the design of cost efficient RDO hardware with high decision quality.

11.6.1 Rate-Distortion Optimized Mode Decision in HM

The mode decision process of HM is shown in Fig. 11.14. HM uses a two-step RDO algorithm for mode decision. Firstly, fast RDO is used for early termination, and then full RDO is used for final decision. To have better mode decision quality and improve coding gain, full RDO is applied among the following aspects:

- The directions and modes chosen by fast RDO.
- The best selected intra directions and inter prediction modes.
- CU depth levels.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00369

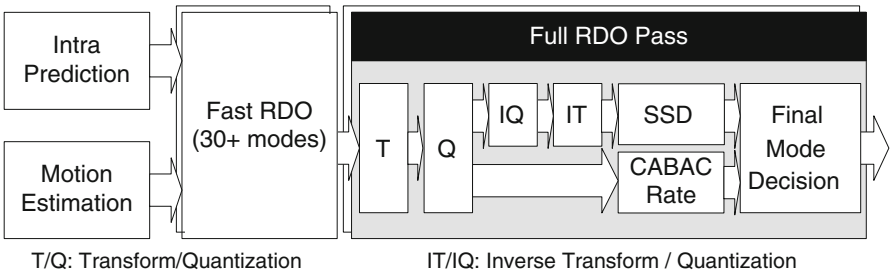


Fig. 11.14 RDO algorithm flow in HM

Firstly, fast RDO is done. To save computation overhead, fast RDO selects several candidates among intra prediction directions and inter prediction motion vectors and modes for each depth level. In fast RDO, the rate is determined by mode bits and motion vectors, and the distortion is calculated by the sum of absolute difference (SAD) or the sum of absolute transformed difference (SATD). Although fast RDO is not accurate, it is still able to prune the less probable cases. Fast RDO selects totally more than 30 modes depending on encoder configurations. Then, the full RDO costs are estimated and compared for these modes. In full RDO process, all the residues are transformed, quantized, inverse quantized, and inverse transformed to produce the reconstructed differences. Distortion is then calculated by the sum of squared difference (SSD). The prediction information and residue coefficients will go through CABAC bit estimator to obtain bit rate if estimated mode is selected. After that, final decision between the modes is made by Lagrangian cost with SSD distortion and estimated CABAC bit rate to optimize the trade-off.

11.6.2 Proposed Hardware RDO Mode Decision Pipeline

In hardware, we also use a hardware-oriented two-step RDO algorithm for mode decision. Figure 11.15 shows the overall RDO mode decision hardware architecture. RDO mode decision requires several major functional units to cooperate. Thus, several CU-level pipeline stages are shown. In the first step, mode pruning is done in intra and IME stages. FME refines all the modes selected by inter motion estimation. After that, full RDO is performed for each mode. A High Complexity Mode Decision (HCMD) hardware consisting of a bit rate estimator and a SSD cost unit is used for each mode that needs full RDO. The final mode is decided by comparing the resulting costs from HCMD hardware in all selected candidates. After that, the context state update for bit rate estimator is performed according to the final modes. More details on the HCMD hardware are provided in Sect. 11.6.4.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00370

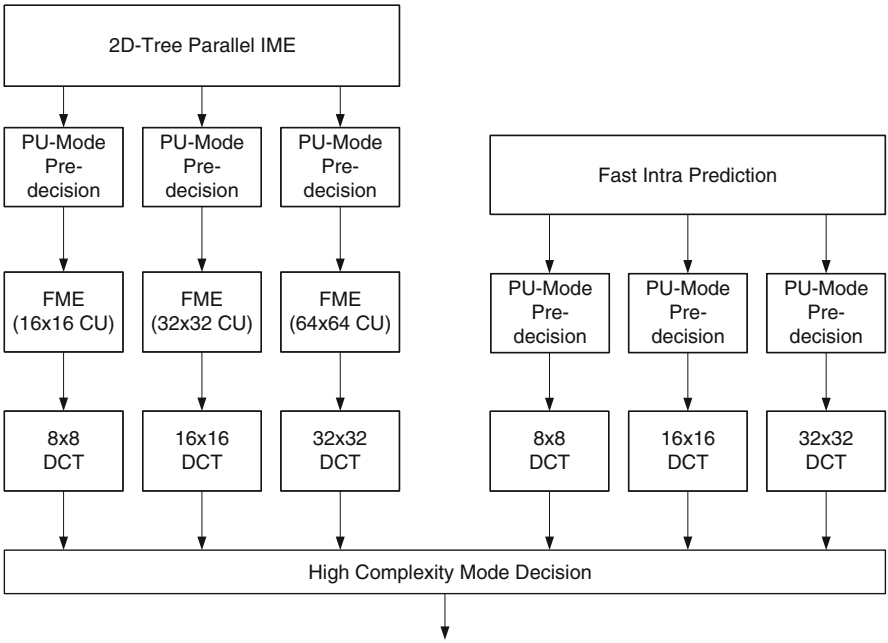


Fig. 11.15 HCMD pipeline architecture block diagram

11.6.3 Hardware-Oriented Two-Step RDO Algorithm

In this section, we present a two-step mode decision flow for hardware. In the literature, various coding tree pruning algorithms are proposed to further reduce the full RDO numbers for computing CU depth [16, 26, 27, 29]. Instead of a hard threshold, a fast CU splitting and pruning scheme based on Bayes decision rules and Gaussian distribution of RD-cost is proposed in [35]. For intra prediction modes, most probable mode (MPM) is derived from neighboring blocks as alternative candidates for full RDO to improve the mode decision quality within the limited number of candidates from rough mode decision [42]. Intra CU depth traversal can also be early terminated by neighboring CU mode and block size relationship between TU and PU [14]. With these early termination methods, only candidates with good enough costs from fast RDO will be selected to go through the full RDO process. The final mode will be chosen from the full RDO result.

The parallelization cost per computation in CABAC hardware is much higher than other modules. This is because most of the CABAC cost is from context memory that changes according to the chosen mode. As a result, it is hardly sharable. Thus, the parallelization cost required to reach the throughput is rather high. Many fast algorithms propose to use fast RDO as the final mode decision. In most of the previous works on H.264/AVC encoder, this method is applied with various fast RDO algorithms. A previous low power encoder [9] in H.264/AVC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00371

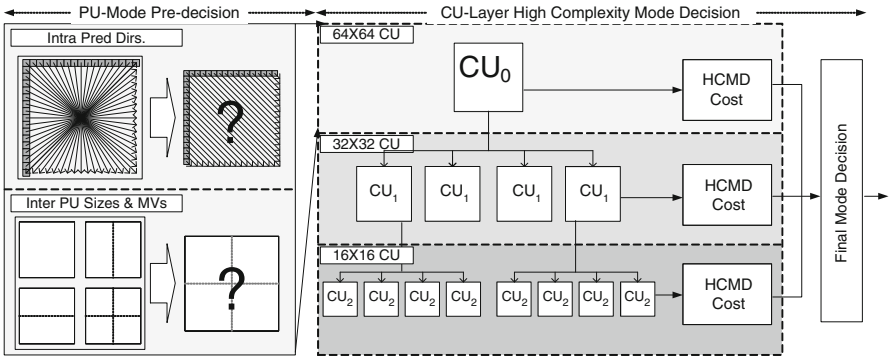


Fig. 11.16 Hardware-oriented two-step RDO algorithm flow

also used a mode pre-decision scheme at IME to reduce the computation for FME. However, the BD rate increase for the fast RDO only method is quite high in HEVC. If we cancel all the full RDO and use only fast RDO in HM, the BD rate increases 10−15 % in intra frames and may even increase to more than 40 % in inter frames. Therefore, it seems that it is quite harmful to eliminate the full RDO completely due to inaccurate prediction of rate-distortion cost. To keep the coding quality while reducing cost for RDO process, hardware encoder should have a more limited number of full RDO, but still keep the important decisions to full RDO.

Figure 11.16 shows the proposed two-step RDO algorithm flow. Since it is expensive to use full RDO for all mode decisions, we use full RDO among the best selected intra directions and inter prediction modes, and among CU depth levels. For each CU depth level, the final direction and mode is decided only by fast RDO. Thus, the number of modes is reduced to one per prediction type and CU depth level. This mode pruning step occurs at intra prediction stage for intra modes and integer motion estimation stage for inter modes. In intra prediction, the distortion cost is SATD, and the rate cost is mode bits. In integer motion estimation, the distortion cost is SAD, and the rate cost is motion vectors difference bits. In the next step, more accurate costs for the selected modes are calculated by HCMD hardware, which performs full RDO. The detail implementation of the HCMD hardware will be discussed in Sect. 11.6.4. After that, final mode is chosen accordingly. With the two-step RDO algorithm, the number for full RDO that needs HCMD hardware is decreased to 6, at the cost of 5.93 % BD-rate increase.

11.6.4 High Complexity Mode Decision

In the previous section, two-step RDO algorithm reduces the number of candidates that require full RDO to 6. However, full RDO is still required to prevent large quality drop. As a result, we still need efficient hardware design to take care of

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00372

full RDO process. This is done in proposed HCMD hardware. HCMD hardware consists of SSD unit and CABAC bit rate estimator. The two parts are discussed in Sects. 11.6.4.1 and 11.6.4.2, respectively.

11.6.4.1 SSD Cost Unit

Since SSD is done only on final mode decision in HCMD and does not require high throughput as SAD/SATD unit in prediction stage, direct implementation is feasible. Consider the following case as an example. If PU-level early mode decision is applied, six modes need to pass through HCMD process. Assume we are encoding 8K UHDTV sequence at 30 fps. The clock rate is set to be 300 MHz. CTU size is 64×64. For each CTU, there will be about 1,200 cycles to process. We may use four multiplier and sum units per mode, and SSD computation for six modes are done in parallel. The total cycles required are 1,024 cycle and the throughput is acceptable.

11.6.4.2 CABAC Bit Rate Estimator

CABAC is the only choice for entropy coding in HEVC because of its coding performance. However, the CABAC has strong sequential dependency and is difficult to parallelize; it also has high implementation cost. In HCMD, multiple instances of CABAC are used for bit estimation. Large area is required if bit estimation is done with CABAC. The major cause of the area is that CABAC uses high number of *contexts* to attain accurate probability estimation. Each *context* stores one *{state, MPS}* pair in memory. The huge amount of *{state, MPS}* memory results in large cost in state stage. Since each CABAC needs to trace state for each mode, multiple instances of CABAC state storage is required. *State stage* occupies most area in CABAC. This is not efficient for implementation.

There are some other methods that use regression-based or table-based methods for prediction. The bit rate can be predicted accurately by table lookup [21]. JCTVC-G763 [1] proposes a table-based CABAC bit counting algorithm. Fractional numbers of bits ranging from 0.008 to 7.497 bits are accumulated according to current state. However, it still relies on the states of CABAC. Thus, it still needs to traverse the states of CABAC and requires separate storage for states of each HCMD mode. The sequential nature of CABAC also poses a limit on the throughput of these bit counters that require CABAC states.

To reduce the cost from CABAC bit estimation, we need to resolve the state issue. We show two hardware-oriented algorithms: bypass-based bit estimation and Context-Fixed Binary Arithmetic Coding (CFBAC) algorithm. For the bypass-based bit estimation, we do not actually do CABAC. We only sum up the bit count output by the binarization process (this is equivalent to coding the bins in bypass mode). Since we do not pass the bitstream to the arithmetic encoder, this technique does not require the state to be stored. Thus, state memory cost is saved in this case. For the CFBAC algorithm, we aim to reduce the state memory cost by sharing the

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00373

Table 11.3 Bit estimation algorithm comparison

Algorithm	JCTVC-G763(HM)	CFBAC	Bypass-based	Fast RDO only
ΔBD-Rate[%] (vs. CABAC)	−0.13	1.14	2.65	48.31

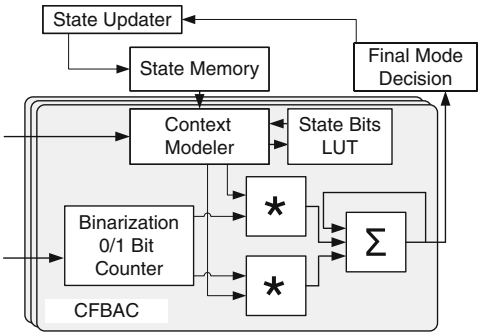


Fig. 11.17 CFBAC architecture block diagram

state memory between modes. The issue for sharing state memory is that states are updated in arithmetic coding process. To eliminate this issue, we use fixed state memory that is not updated during bit estimation. However, if the states used in the bit estimation are too different from the actual states, the bit estimation will not be accurate enough and cause low quality decision in HCMD. Hence, we keep the context fixed at a CTU-level. bit rate increase for this scheme is limited [17]. The states are the same in CABAC at the beginning of the CTU. During the bit estimation process inside CTU, the states are not updated. After the final mode decision is made, the bits for the selected mode are traversed and the final states are updated. For more simplification, we also uses bit estimation table in JCTVC-G763 [1] for bits look-up instead of arithmetic encoder in context-fixed scheme.

For quality comparison, the BD-rate differences vs. CABAC-based bit estimation are shown in Table 11.3. HM takes JCTVC-G763 as the default fast bit estimator. As we can see, the quality drop would be high if all the mode decision is done only by fast RDO. For bypass-based method, the quality loss is moderate and hardware cost is low. If more accurate result is required, CTU-based CFBAC can be used.

The hardware architecture for CFBAC is shown in Fig. 11.17. Since the states are fixed, all the MPS and LPS coded using the same type of context share the same probability. For every '1' bin and '0' bin in the same context, the bits produced is a fixed number B_0 and B_1 according to the JCTVC-G763 look-up table, respectively. So we need only to count the number of '1' bins and '0' bins for bit rate estimation. We modify the binarization process and produce the 1's count C_1 and 0's count C_0. The bit rate can be estimated according to Eq. (11.1).

$$F_{bits}(input) = \sum_{\forall n \in contexts} B_0(n) * C_0(n) + B_1(n) * C_1(n) \tag{11.1}$$

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00374

The corresponding architecture is shown in Fig. 11.17. It does not need true CABAC architecture. Instead, it only needs binarization and context part. Additional lookup table is placed for bin-to-bits conversion. The corresponding number of bits for 1's and 0's depend on the CTU-based states, which are shared in the global state memory. Since the state memory is not changed in the CFBAC, the content of each instances of CFBAC is the same. So we do not need to keep a separate copy for each CFBAC instances. Multiple CFBAC may share the same state memory.

With this architecture, most of the cost from state memory are saved with only 1.25 % BD rate increase compared to HM, and has higher throughput. The context memory for CFBAC can be further saved by sharing the state memory with entropy encoder if the entropy encoder and mode decision engine operate on the same frame.

11.6.4.3 Final Mode Decision and State Update

After the bit count is estimated, mode decision is performed. However, we still need to update the context memory according to the chosen mode for the bit estimation of the next CU. This is done by traversing the final mode bits. Since we only need to know the final states and do not need to do arithmetic coding, we can simplify the original CABAC process. We may use an $nM1L$ architecture for fast state transition as follows. For every MPS, the state is always increased by one until the top state (63) is reached, we only need to count the number of MPS for state prediction. For LPS, a $64 - entry$ table lookup is required. As such, we may process n MPS and one LPS at one time by one table lookup. The speedup is n times compared to CABAC state architecture, where n depends on the bitstream. This process is fast and can also be cascaded for higher performance.

11.7 In-Loop Filters

For in-loop filters, there are two filters in HEVC, deblocking filter and sample adaptive offset (SAO) filter. Compared to H.264/AVC, deblocking filter in HEVC is simplified. Deblocking may be divided into two passes. Each direction (horizontal or vertical) is done in one pass. On the other hand, the SAO filter is a new coding tool in HEVC. It collects statistics of pixel distortion and minimizes the difference between input samples and reconstructed samples by adding an adaptive offset. SAO filter types can be chosen from Edge Offset (EO), Band Offset (BO), or unchanged (OFF). EO performs pixel classification based on edge direction/shape. BO is based on pixel level. If the pixel is not suitable for SAO, it can be marked as unchanged. SAO encoding is more complex than deblocking. It consists of offset derivation stage, and filtering stage. Offset derivation stage collects required statistics information from original and reconstructed CTU. After that, offsets and types are decided with the statistics information. Then, the filtering stage will perform offset filtering according to the offsets and types. In HM, the two in-loop filters are processed in serial. However, this will cause pipeline to be even longer. Since SAO requires

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00375

both the original and reconstructed CTUs, each increased stage would require two additional CTU-sized buffers. To reduce the required number of stages, we may do the deblocking filter and SAO filter together. To remove the latency, SAO coefficient derivation can use the non-deblocked reconstructed CTU in place of the deblocked one [25]. This causes minimal quality loss since deblocking and SAO target different artifacts. To combine the dataflow of the two filters, the deblocking first pass is done along with SAO coefficient derivation in parallel. The reconstructed CTU buffer supplies data for both modules in parallel. After that, the second deblocking pass is performed; SAO filtering is done right after the deblocking filter. In this way, the loop filtering can be performed efficiently.

11.8 Entropy Coding

Entropy coding is used to remove redundancy that is not eliminated by prediction tools. It uses the probability distribution of the syntax elements. It also plays an important role in video coding. CABAC is adopted in HEVC as the default entropy coding tool since it achieves 6.1–7.6 % bit-rate saving over Context-Based Adaptive Variable Length Coding (CAVLC) [17]. While CABAC provides high coding efficiency, its process exhibits highly complex bin-to-bin data dependencies. As a result, CABAC encoder is usually one of the most critical throughput bottlenecks in the whole video encoder. But compared to H.264/AVC, there are many methods in HEVC that make parallel processing of CABAC possible. In this section, the high throughput CABAC design and parallelism design of CABAC are discussed.

For CABAC hardware design, there are two-stage [3], three-stage [32], and four-stage [12, 19, 20, 34] pipelines. Four-stage is mostly used in recent high throughput designs. Figure 11.18 shows the four-stage overall CABAC architecture for H.264/AVC. Syntax finite-state machine (FSM) controls the coding order of the syntax elements. The data access module prepares the required data for binarizer and context modeler. Binarizer will convert original syntax elements to binary streams. Context modeler determines the next context states to be used. After that, binary arithmetic coding is applied.

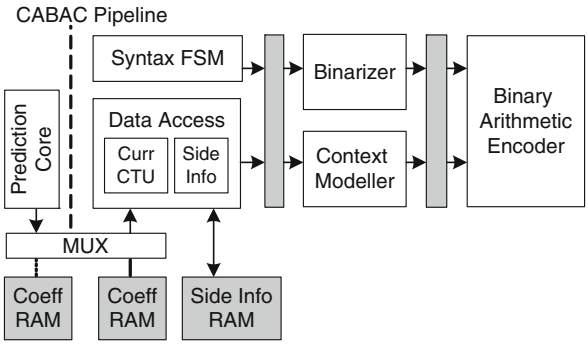


Fig. 11.18 Overall CABAC architecture for H.264/AVC

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00376

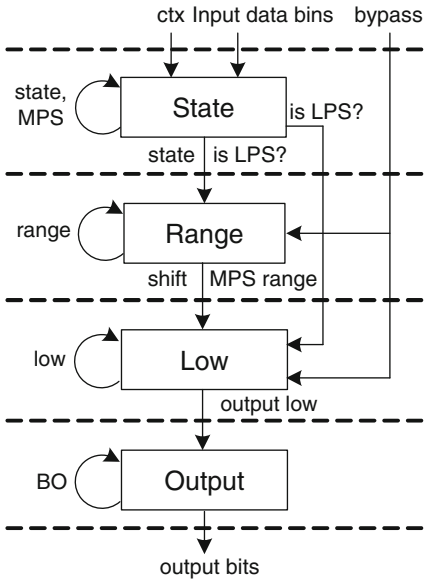


Fig. 11.19 Basic one-bin CABAC pipeline scheme

Typical arithmetic coder can be further partitioned into four main stages: *State*, *Range*, *Low* and *Output*. *State* stage will update MPS and state as shown in Fig. 11.19. *Range* and *Low* stage will update the range and low values for arithmetic encoding. *Output* stage will be in charge of outputting the bitstream. Normalization is performed on range and low after encoding each bin so that they can be represented with a fixed 9-bit precision. We can see data dependencies between the four blocks. With this architecture, one bin per cycle is achieved. To achieve better throughput, pre-normalization circuit may be used to reduce normalization critical path [44]. In HEVC, more than 20 % of the bins are bypass bins. Since the range update circuit and the context model are not affected in bypass coding, a bypass bin spitting (BPBS) scheme can be applied to split the process from the bin stream and remerge into the bitstream before the low update stage [44].

For high resolution applications, the throughput of one-bin CABAC is not enough. Because of CABAC data dependencies, it is difficult to add more pipeline stages in CABAC. As a result, techniques to encode more than one bin in a cycle may be needed. Prior related work includes two-bin [3, 19, 20, 34] and multi-bin [12, 44] arithmetic encoder. One method of multi-bin CABAC is cascading. By cascading the State, Range, Low, and Output circuits and by using a state forwarding circuit in State stage, a CABAC engine with multi-bin per cycle is achieved as shown in Fig. 11.20a, b. Another method for multi-bin CABAC is the state dual-transition (SDT) approach [44]. It combines two state transition tables into one at the cost of a bigger table. Then, each stage may process two bins per cycle. SDT can also be combined with cascading techniques.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00377

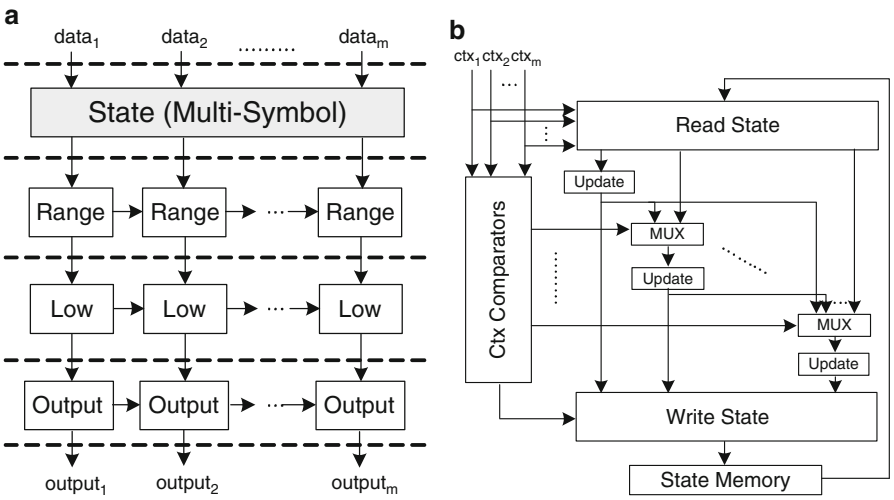


Fig. 11.20 Multi-bin (**a**) overall block diagram (**b**) state forwarding circuit

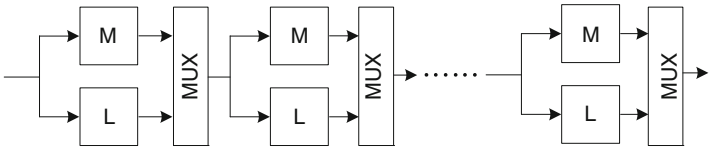


Fig. 11.21 Simplified architecture of multi-bin binary arithmetic encoder

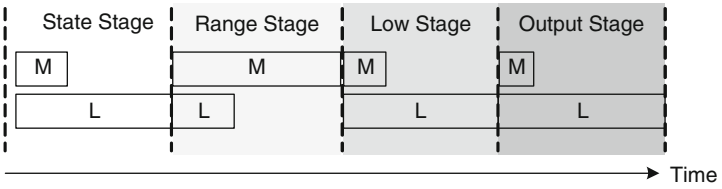


Fig. 11.22 Branch imbalance of four-stage pipeline architecture in H.264/AVC [11]

The effectiveness of cascading technique is still limited because of growing delay in state forwarding circuit as the number of cascading stages increase. The operating frequency will be reduced if too many cascading stages are used. For higher throughput, a ML-decomposed architecture may be applied as follows [11]. We can observe from the CABAC pipeline that the complexity for processing the MPS and LPS in each stage is different. We can divide the processing into two parts: M for MPS and L for LPS for timing analysis. A typical multi-bin CABAC architecture is shown in Fig. 11.21. After analysis, we can observe the imbalance as indicated in Fig. 11.22. The MPS and LPS have different latencies. To have higher throughput, we may divide the arithmetic into two separate coders.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00378

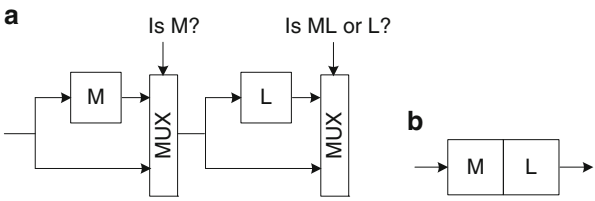


Fig. 11.23 (**a**) ML cascade architecture (**b**) Simplified representation [11]

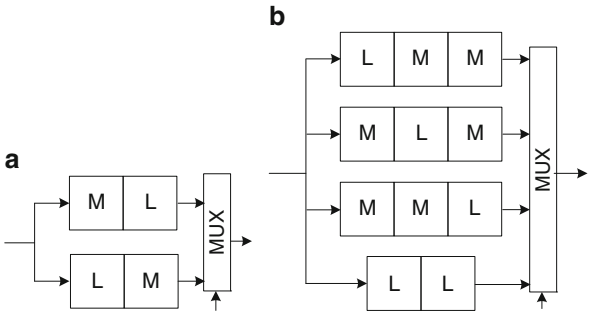


Fig. 11.24 Examples of throughput-selection architecture with balanced critical paths [11]

For one MPS encoder and one LPS encoder, the throughput per cycle in traditional architecture is always one bin only. But it is possible to fully utilize both encoders to code two bins in one cycle. The corresponding architecture can be easily configured as in Fig. 11.23a and in a more simplified form as in Fig. 11.23b. Now, two bins are simultaneously checked in one cycle. If the two bins are MPS and LPS in order, both M coder and L coder are active. If the two bins are both MPS, only the M coder is active to encode the first bin, and the second bin will be coded in the next cycle.

Similarly, when the first bin is LPS, only the L coder is active. Therefore, the throughput per cycle is improved from one bin to one or two bin. Although the critical path becomes longer, the overhead is moderate because the original critical paths of M and L coder are quite unbalanced as shown in Fig. 11.22, and thus complimentary to each other. To make best use of the timing slack, throughput-selection architecture may be applied to increase the throughput while maintain similar critical path between different paths. Therefore, the design strategy is to make the critical paths of all choices balanced. Both *{ML,LM}* and *{MML,MLM,LMM,LL}* in Fig. 11.24 are good examples of throughput-selection architecture with balanced critical paths. The number of choices can be fine-tuned to fit the target throughput. If higher throughput is required, throughput-selection architecture can be further cascaded. In addition, multiple M stages can be shared by forwarding the first M result to alternative path in the throughput selection circuit [44].

For high bin rate requirement, using a single CABAC engine in some situations cannot achieve the required bin rate. HEVC in such circumstances has provided several parallelization schemes at the cost of marginal bit rate increase. HEVC provides wavefront-parallel CABAC, tiles, and slices with different constraint on

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00379

Table 11.4 Comparison of H.264/AVC and HEVC encoders

	ISSCC'09[18]	VLSIC'12[43]	This work
Resolution	4096x2160@24fps	7680x4320@60fps	8192x4320@30fps
Throughput	212Mpixels/s	1991Mpixels/s	1062Mpixels/s
Standard	H.264 High @ Level 5.1	H.264 Intra	HEVC
Search range	[−255,+255]/[−255,+255]	N/A	[−512,+511]/[−128,+127] (Predictor Centered)
Technology	TSMC 90nm	e-Shuttle 65nm	TSMC 28nm HPM
Core size	3.95x2.90mm^2	3.95x2.90mm^2	5x5mm^2
Gate count	1732K	678.8K	8350K
Power	522mW@280MHz	139.9mW@280MHz	708mW@312MHz

Table 11.5 Summary of modifications

Module	Modifications	ΔBD-Rate[%]	Δ BD-PSNR[dB]
Intra	1. 12-Candi. Fast Intra Prediction 2. Skip I64 3. Hybrid Open-Close Loop	1.03	−0.03
Inter	1. Parallel-PU IME 2. Two-AMVP Coarse-fine Search 3. 3-bit Pixel Truncation 4. Quarter Sub-sampling 5. 25-Candi. Central Quarter FME	6.08	−0.19
Transform	1. Fixed Transform Size	3.02	−0.08
RDO	1. PU-level Early Mode Decision 2. CFBAC Bit Estimator	7.18	−0.20

CABAC dependency. Multiple parts of frame (i.e. multiple CTUs) can be coded at the same time if these configurations are enabled. In addition, multiple sets of CABAC engines can be used to make the utilization even higher.

11.8.1 Implementation Results for Encoder Test Chip

An HEVC encoder test chip capable of encoding 8K UHDTV is implemented in [36]. The encoder is designed based on HM4. The modifications relative to HM is summarized in Table 11.5. The primary gate count of the encoder is listed in Table 11.6. Note that frame-level loop filter in HM4 is implemented in this encoder; however, as frame-level loop filter is not required in the final standard, the gate count would be significantly reduced for an HEVC-compliant implementation. Comparisons to previous AVC encoders is shown in Table 11.4. The total bandwidth is 6.80 GB/s. Compared to Ding's previous H.264/AVC encoder in ISSCC'09 [18], the resolution is four times higher, but the bandwidth usage is only increased by 37 %. To compare with H.264/AVC encoders, the rate-distortion curves for HM4, JM (H.264 reference software), and the presented hardware encoder in encoding 8K

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00380

Table 11.6 Module gate count

Module	Gate count [kGates]
Intra	1148
Inter	2291
Transform	1135
On-chip buffer for prediction	1404
Others*	2372

* Including HM4 frame-level loop filters, which is removed from final standards

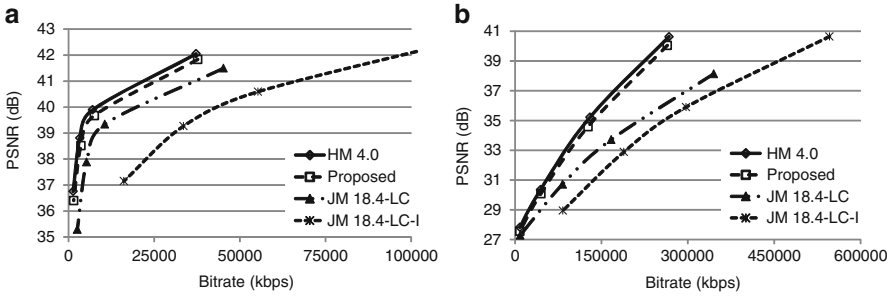


Fig. 11.25 RD curve comparison for 8K sequence. Both sequences are cropped to 2,560 × 1,600 and converted to 8 bit per channel. JM results are also included to show the coding gain over the previous H.264/AVC encoders, where LC stands for low complexity mode decision (i.e. fast RDO only), and I stands for intra mode only. (**a**) Steam locomotive train. (**b**) Nebuta festival

sequences are shown in Fig. 11.25. Both 8K sequences are cropped to 2,560×1,600-8bit in this test. The test condition is low delay P, with a maximum of two reference frames and maximum CU depth of three. Numerically, average 22.6 % BD-rate increase is shown compared to HM4. The BD rate increase is more in low bit rate region and less in high bit rate region. Encoding quality for JM is significantly lower than that for HM4. The RD-curve for the presented HEVC encoder hardware, in comparison, is close to the one for HM4. With proper selection of architecture, a video encoder can be designed to achieve both high coding efficiency and real-time high resolution encoding with reasonable hardware cost.

11.9 Conclusion

In this chapter, we have introduced a hardware encoder design for HEVC. Key design issues in system pipeline, module level design, and high complexity mode decision that supports full RDO in hardware have been discussed. A test chip which supports 8K UHDTV real-time encoding in HEVC is also presented. Although HEVC is a complex standard, we can still achieve efficient implementation with proper design of algorithm and architecture. With the techniques presented in this chapter, we show that HEVC can be used for real-time encoding for ultra high resolution applications.

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00381

References

1. Bossen F (2011) CE1: Table-based bit estimation for CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G763, Geneva, Nov. 2011
2. Budagavi M, Sze V (2012) Unified forward+inverse transform architecture for HEVC. In: IEEE international conference on image processing (ICIP), pp 209–212, 2012
3. Chang YW, Fang HC, Chen LG (2004) High performance two-symbol arithmetic encoder in JPEG 2000. In: Proceedings of ISCE, 2004
4. Chang HC, Chen JW, Su CL, Yang YC, Li Y, Chang CH, Chen ZM, Yang WS, Lin CC, Chen CW, Wang JS, Guo JI (2007) A 7mW-to-183mW dynamic quality-scalable H.264 video encoder chip. In: IEEE international solid-state circuits conference (ISSCC), 2007
5. Chang HC, Chen JW, Su CL, Yang YC, Li Y, Chang CH, Chen ZM, Yang WS, Lin CC, Chen CW, Wang JS, Guo JI (2008) A 242mW 10mm2 1080p H.264/AVC high-profile encoder chip. In: IEEE international solid-state circuits conference (ISSCC), 2008
6. Chen TC, Huang YW, Chen LG (2004) Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC. In: IEEE international conference on acoustics, speech, and signal processing (ICASSP), 2004
7. Chen C-Y, Chien SY, Huang YW, Chen TC, Wang TC, Chen LG (2006) Analysis and architecture design of variable block size motion estimation for H.264/AVC. IEEE Trans Circuits Syst Part I 53(3):578–593
8. Chen CY, Huang CT, Chen LG (2006) Level C+ data reuse scheme for motion estimation with corresponding coding orders. IEEE Trans Circuits Syst Video Technol 16(4):553–558
9. Chen TC, Chen YH, Tsai CY, Tsai SF, Chien SY, Chen LG (2007) 2.8 to 67.2mW low-power and power-aware H.264 Encoder for Mobile Applications. In: IEEE symposium on VLSI circuits (VLSIC), 2007
10. Chen TC, Chen YH, Tsai SF, Chien SY, Chen LG (2007) Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC. IEEE Trans Circuits Syst Video Technol 17:568–577
11. Chen YJ, Tsai CH, Chen LG (2007) Novel configurable architecture of ML-decomposed binary arithmetic encoder for multimedia applications. In: International symposium on VLSI design, automation and test (VLSI-DAT), 2007
12. Chen YH, Chuang TD, Chen YJ, Li CT, Hsu CJ, Chien SY, Chen LG (2008) An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip. In: IEEE symposium on VLSI circuits (VLSIC), 2008
13. Chen YH, Chen TC, Tsai CY, Tsai SF, Chen LG (2008) Data reuse exploration for low power motion estimation architecture design in H.264 encoder. J Signal Process Syst 50(1):1–17
14. Cho S, Kim M (2013) Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding. IEEE Trans Circuits Syst Video Technol 23(9):1555–1564
15. Choi K, Jang ES (2012) Early TU decision method for fast video encoding in high efficiency video coding. Electron Lett 48(12):689–691
16. Correa G, Assuncao P, Agostini L, da Silva Cruz LA (2011) Complexity control of high efficiency video encoders for power-constrained devices. IEEE Trans Consum Electron 57(4):1866–1874
17. Davies T, Fuldseth A (2011) Entropy coding performance simulations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F162, Torino, July 2011
18. Ding LF, Chen WY, Tsung PK, Chuang TD, Chiu HK, Chen YH, Hsiao PH, Chien SY, Chen TC, Lin PC, Chang CY, Chen LG (2009) A 212MPixels/s 4096x2160p multiview video encoder chip for 3D/quad HDTV applications. In: IEEE international solid-state circuits conference (ISSCC), 2009
19. Dyer M, Taubman D, Nooshabadi S (2004) Improved throughput arithmetic coder for JPEG2000. In: IEEE international conference on image processing (ICIP), 2004
20. Flordal O, Wu D, Liu D (2006) Accelerating CABAC encoding for multi-standard media with configurability. In: Proceedings of IPDPS, 2006

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00382

21. Hahm J, Kyung CM (2010) Efficient CABAC rate estimation for H.264/AVC mode decision. IEEE Trans Circuits Syst Video Technol 20(2):310–316
22. Hsu MY, Chang HC, Wang YC, Chen LG (2001) Scalable module-based architecture for MPEG-4 BMA motion estimation. In: IEEE international symposium on circuits and systems (ISCAS), 2001
23. Huang YW, Chen TC, Tsai CH, Chen CY, Chen TW, Chen CS, Shen CF, Ma SY, Wang TC, Hsieh BY, Fang HC, Chen LG (2005) A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications. In: IEEE international solid-state circuits conference (ISSCC), 2005
24. Huang YW, Chen CY, Tsai CH, Shen CF, Chen LG (2006) Survey on block matching motion estimation algorithms and architectures with new results. J VLSI Signal Process Syst 42(3):297–320
25. Kim W-S (2012) AhG6: SAO parameter estimation using non-deblocked pixels, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0139, Stockholm, July 2012
26. Kim J, Yang J, Lee H, Jeon B (2011) Fast intra mode decision of HEVC based on hierarchical structure. In: International conference on information, communications and signal processing (ICICS), 2011
27. Kim J, Jeong S, Cho S, Choi JS (2012) Adaptive Coding Unit early termination algorithm for HEVC. In: IEEE international conference on consumer electronics (ICCE), 2012
28. Li F, Shi G, Wu F (2011) An efficient VLSI architecture for 4x4 intra prediction in the High Efficiency Video Coding (HEVC) standard. In: IEEE international conference on image processing (ICIP), pp 373–376, 2011
29. Ma S, Wang S, Wang S, Zhao L, Yu Q, Gao W (2013) Low complexity rate distortion optimization for HEVC. In: Data compression conference (DCC), 2013
30. McCann K, Bross B, Han WJ, Kim IK, Sugimoto K, Sullivan GJ (2013) High efficiency video coding (HEVC) test model 12 (HM 12) encoder description, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-N1002, Vienna, July 2013
31. Meher PK, Park SY, Mohanty BK, Lim KS, Yeo C (2014) Efficient integer DCT architectures for HEVC. IEEE Trans Circuits Syst Video Technol 24(1):168–178
32. Osorio RR, Bruguera JD (2004) Arithmetic coding architecture for H.264/AVC CABAC compression system. In: Euromicro symposium on digital system design, 2004
33. Palomino D, Sampaio F, Agostini L, Bampi S, Susin A (2012) A memory aware and multiplierless VLSI architecture for the complete Intra Prediction of the HEVC emerging standard. In: IEEE international conference on image processing (ICIP), 2012
34. Pastuszak G (2004) A high-performance architecture of arithmetic coder in JPEG2000. In: Proceedings of ICME, 2004
35. Sinangil M, Sze V, Zhou M, Chandrakasan A (2013) Cost and coding efficient motion estimation design considerations for high efficiency video coding (HEVC) standard. IEEE J Sel Top Signal Process 7(6):1017–1028
36. Tsai SF, Li CT, Chen HH, Tsung PK, Chen KY, Chen LG. A 1062Mpixels/s 8192x4320p high efficiency video coding (H.265) encoder chip. In: Symposium on VLSI circuits (VLSIC), 2013
37. Teng SW, Hang HM, Chen YF. Fast mode decision algorithm for Residual Quadtree coding in HEVC. In: IEEE visual communications and image processing (VCIP), 2011
38. Tuan JC, Chang TS, Jen CW (2002) On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture. IEEE Trans Circuits Syst Video Technol 12(1): 61–72
39. Tsung PK, Chen WY, Ding LF, Tsai CY, Chuang TD, Chen LG (2009) Single-iteration full-search fractional motion estimation for quad full HD H.264/AVC encoding. In: IEEE international conference on multimedia and expo (ICME), 2009

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00383

40. Tsung PK, Chen WY, Ding LF, Chien SY, Chen LG (2009) Cache-based integer motion/disparity estimation for quad-HD H.264/AVC and HD multiview video coding. In: IEEE international conference acoustics, speech, and signal processing (ICASSP), 2009
41. Zhang J, Dai F, Ma Y, Zhang Y (2013) Highly parallel mode decision method for HEVC. In: Picture coding symposium (PCS), 2013
42. Zhao L, Zhang L, Ma S, Zhao D (2011) Fast mode decision algorithm for intra prediction in HEVC. In: IEEE visual communications and image processing (VCIP), 2011
43. Zhou D, He G, Fei W, Chen Z, Zhou J, Goto S (2012) A 4320p 60fps H.264/AVC intra-frame encoder chip with 1.41Gbins/s CABAC. In: IEEE symp. VLSI circuits (VLSIC), 2012
44. Zhou J, Zhou D, Fei W, Goto S (2013) A high-performance CABAC encoder architecture for HEVC and H.264/AVC. In: International conference on image processing (ICIP), 2013
45. Zhu J, Liu Z, Wang D (2013) Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC Codec. In: IEEE international symposium on circuits and systems (ISCAS), 2013

IPR2024-01008
Patent Owner's Ex. 2015
Ex. 2015.00384