Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11
1st Meeting: Dresden, DE, 15-23 April, 2010

Document: JCTVC-A124

Title: **Samsung's Response to the Call for Proposals on Video Compression Technology**

Status: Input Document to JCT-VC

Purpose: Proposal

Author(s) or Contact(s):

Ken McCann — ken@zetacast.com, +44 1962 625347
Woo-Jin Han — Tel: Email: wjhan.han@samsung.com, +82 31 279 8831
Il-Koo Kim — ilkoo.kim@samsung.com

Source: Samsung Electronics Co., Ltd. and British Broadcasting Corporation

---

Abstract

This proposal is Samsung's response to the Call for Proposals (CfP) on video compression technology, jointly issued by ITU-T SG16 Q.6 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG). It has been produced in collaboration with the British Broadcasting Corporation. The goal of this proposal is to provide a video compression technology which has significantly higher compression capability than the state-of-the-art H.264/AVC standard, especially for high-definition (HD) video contents. To achieve this goal, a number of new algorithmic tools are proposed covering several aspects of video compression technology. These include a general structure for representation of video content, inter/intra prediction, in-loop filtering, and entropy coding. When all the proposed algorithmic tools are used, the proposed video codec achieves approximately 40% bit-saving on average compared to H.264/AVC in both constraint set 1 and 2 configurations. The average decoding time for the proposed codec is about 10% less than for JM17.0.

SNAP-1011
IPR2025-01263

Contents

Abstract....1
Glossary of terms used in this proposal....4
1 Introduction....5
2 Algorithm description....5
2.1 Overview....5
2.2 Unit definition....7
2.2.1 Coding unit (CU)....7
2.2.2 Prediction unit (PU)....8
2.2.3 Transform unit (TU)....9
2.2.4 Relationship between CU, PU and TU....10
2.2.5 Size-independent syntax representation....10
2.3 Motion representation....11
2.3.1 Asymmetric motion partition (AMP)....11
2.3.2 Advanced motion vector prediction (AMVP)....11
2.3.3 Improved skip and direct mode....12
2.3.4 DCT-based interpolation filter (DIF)....12
2.3.4.1 DCT basis decomposition....12
2.3.4.2 Example of filter coefficients....13
2.3.4.3 Low complexity optimization....14
2.3.5 High accuracy motion (HAM)....14
2.4 Intra-frame prediction....15
2.4.1 Arbitrary directional intra (ADI)....16
2.4.2 Multi-parameter intra (MPI)....18
2.4.3 Color Component correlation based prediction (CCCP)....19
2.4.4 Pixel based template matching (PTM)....20
2.5 Spatial transforms....20
2.5.1 Large transform (16x16, 32x32 and 64x64)....20
2.5.2 Rotational transform (ROT)....21
2.5.3 Logical transform (LOT)....22
2.5.4 Quantization....23
2.6 In-loop filtering....23
2.6.1 Deblocking filter....23
2.6.2 CU-synchronized adaptive loop filter (ALF)....23
2.6.3 Extreme correction (EXC) and band correction (BDC)....24
2.6.4 Content adaptive dynamic range (CADR)....24
2.7 Entropy coding....25
2.7.1 Syntax-based context-adaptive binary arithmetic coding (SBAC)....25
2.7.2 Adaptive coefficient scanning (ACS)....26
3 Compression performance discussion....27
3.1 Objective versus subjective compression performance....27
3.2 Constraint set 1 configuration relative to Alpha anchor....27
3.3 Constraint set 2 configuration relative to Beta and Gamma anchors....28
4 Complexity analysis....30
4.1 Encoding time and measurement methodology....30
4.2 Decoding time and measurement methodology....31
4.3 Expected memory usage of encoder....32
4.4 Expected memory usage of decoder....32
4.5 Complexity characteristics of encoder motion estimation and motion segmentation selection....32
4.5.1 Motion estimation....32
4.5.2 Motion segmentation selection....32
4.6 Complexity characteristics of decoder motion compensation....33
4.6.1 Interpolation process....33
4.6.2 Number of motion compensations....33
4.7 Complexity characteristics of encoder intra-frame prediction type selection....33
4.8 Complexity characteristics of decoder intra-frame prediction operation....33
4.9 Complexity characteristics of encoder transforms and transform type selection....34

4.10 Complexity characteristics of decoder inverse transform operations 34
4.10.1 Large size DCT 34
4.10.2 ROT 35
4.11 Complexity characteristics of encoder quantization and quantization type selection 35
4.12 Complexity characteristics of decoder inverse quantization 35
4.13 Complexity characteristics of encoder in-loop filtering type selection 36
4.13.1 Adaptive loop filter 36
4.14 Complexity characteristics of decoder in-loop filtering operations 36
4.14.1 Deblocking filter 36
4.14.2 Adaptive loop filter 36
4.14.3 Extreme correction and band correction 37
4.15 Complexity characteristics of encoder entropy coding type selection 37
4.16 Complexity characteristics of decoder entropy decoding operation 37
4.16.1 Syntax-based context-adaptive binary arithmetic coding 37
4.16.2 Adaptive coefficient scanning 37
4.17 Degree of capability for encoder parallel processing 37
4.18 Degree of capability for decoder parallel processing 38
4.19 Requirement for encoder frame-level multi-pass processing 38
5 Algorithm characteristics 38
5.1 Random access characteristics 38
5.2 Delay characteristics 38
6 Software implementation description 39
7 Closing remarks 39
8 Patent rights declaration(s) 39
References 39

Glossary of terms used in this proposal

ACS	adaptive coefficient scanning
ADI	arbitrary directional intra
ALF	adaptive loop filter
AMP	asymmetric motion partition
AMVP	advanced motion vector prediction
BDC	band correction
CADR	content-adaptive dynamic range
CCCP	color component correlation based prediction
CU	coding unit
DIF	DCT-based interpolation filter
EXC	extreme correction
HAM	high accuracy motion
IBDI	internal bit-depth increase
LCU	largest coding unit
LOT	logical transform
MPI	multi-parameter intra
PTM	pixel-based template matching
PU	prediction unit
ROT	rotational transform
SBAC	syntax-based context-adaptive binary arithmetic coder
SCU	smallest coding unit
SPS	sequence parameter set
TU	transform unit

1 Introduction

With the ever increasing popularity of high-definition (HD) video content, video compression technologies which can provide higher coding efficiency than existing video coding standards have received increased attention. Recently, ITU-T SG16 Q.6 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG) have decided to create a Joint Collaborative Team on Video Coding (JCT-VC), and have jointly issued a Call for Proposals (CfP) on video compression technology. This proposal is Samsung's response to the CfP, produced in collaboration with the BBC. It aims to provide a video compression technology which has significantly higher compression capability than the state-of-the-art H.264/AVC standard, especially for HD video content. To achieve this goal, a number of new algorithmic tools are proposed, covering many aspects of video compression technology. These include a general structure for the representation of video content, inter/intra prediction, in-loop filtering, and entropy coding.

For efficient and flexible representation of video content with various resolutions, the representation introduced in this proposal is based on a triplet of coding unit (CU), prediction unit (PU), and transform unit (TU). CU is the basic unit of compression; CU sizes larger than 16x16, e.g. 32x32, 64x64 or even 128x128 are supported in order to compress HD video content effectively. CU is broadly analogous to the concept of macroblock, but is much more flexible. PU is the unit of inter/intra prediction and there can be multiple PUs in a single CU. To code irregular image patterns effectively in a video frame, asymmetric motion partition (AMP) for inter prediction is also supported. TU is the unit of transform, and can include one or more PUs. In addition to conventional 4x4 and 8x8 transforms, 16x16, 32x32, and 64x64 fast integer transforms are supported for TUs.

To improve inter prediction performance, a number of novel algorithmic tools are proposed. Advanced motion vector prediction (AMVP) is proposed, for the efficient representation of motion information. Sub-pixel high accuracy motion (HAM), with resolution down to 1/12 pixel, is also supported along with new 1/4-pel based interpolation filters designed separately for each phase and replacing the concatenated half-band and bilinear interpolation architecture for H.264/AVC. All filters have been designed using a common DCT interpolation filter (DIF) method.

To improve the performance of intra prediction, four new intra coding tools are included: arbitrary directional intra (ADI), pixel based template matching (PTM), color component correlation based prediction (CCCP), and multi-parameter intra (MPI). A rotational transform (ROT) tool, which can adaptively compact the energy into low frequency components, is also proposed. Using these algorithmic tools, prediction patterns which cannot be generated from directional pattern extrapolation or simple calculation from neighboring pixels can be provided; this leads to reduced energy of the residual signal and hence to improved coding performance.

In this proposal, several in-loop filters are combined to reduce the distortion between the original and the reconstructed video frames. In addition to a modified H.264/AVC deblocking filter for the CU/PU/TU structure, an adaptive loop filter (ALF) is applied to reduce the average distortion of the reconstructed frame, and then extreme correction (EXC) is used to reduce the distortion for local regions. Content adaptive dynamic range adaptation (CADR) is also performed to mitigate the rounding effects. All of the in-loop filtering operations are performed adaptively with respect to the flexible unit representation.

A syntax-based context-adaptive binary arithmetic coder (SBAC) is proposed for the entropy coding algorithm. SBAC is an adaptive binary arithmetic coding method with context models, which can provide high coding efficiency for various syntax elements with varying statistical properties. In order to increase the efficiency of entropy coding of transform coefficients, adaptive coefficient scanning (ACS) is used, which explicitly specifies the scanning order for each TU.

When all the proposed algorithmic tools are used, the proposed video codec achieves approximately 40% bit-saving on average compared to H.264/AVC in both constraint set 1 and 2 configurations. Furthermore, the bit-savings tends to be larger when the spatial resolution of the video increases, which is a good property according to the major goal of the HVC standard.

2 Algorithm description

2.1 Overview

In this proposal, new algorithmic tools are presented to exploit the properties of high resolution video. Highlighted features of proposed codec are as follows.

- **Flexible size unit representation**: the proposal separately defines three block concepts: coding unit (CU), prediction unit (PU) and transform unit (TU). After the size of largest coding unit (LCU) and the hierarchical depth of CU have been defined, the overall structure of codec is characterized by the various sizes of CU, PU and TU in a recursive manner. This allows the proposed codec to be readily adapted for various kinds of content, applications, or devices that have different capabilities/resources.

- **Size-independent syntax representation:** while block level syntax such as coded block pattern and intra prediction mode are coded differently depending upon the block sizes in H.264/AVC, the proposed codec employs one common syntax representation for all CU sizes, which reduces complexity and improves clarity.
- **Support of large and asymmetric motion partitions:** larger PUs than 16x16 are supported. Asymmetric motion partition (AMP) is also supported, to increase the performance for irregular image patterns.
- **Support of higher motion accuracy than 1/4 pel with new interpolation filter**: high accuracy motion (HAM) is supported with a new DCT-based interpolation filter (DIF). Motion vector refinement is introduced to obtain high accuracy such as 1/12 pixel.
- **Support of large integer transforms**: in addition to conventional 4x4 and 8x8 transform, fast integer realizations of 16x16, 32x32 and 64x64 transforms are proposed.
- **New rotational transform**: a new supplementary rotational transform (ROT) is proposed to encode high energy residual information more efficiently.
- **New motion vector prediction method**: advanced motion vector prediction (AMVP) is utilized to find the best motion vector predictor among the various PU combinations
- **New in-loop filtering methods**: several in-loop filters are combined to reduce the reconstruction distortion. The H.264/AVC deblocking filter has been modified to make it suitable for the hierarchical CU/PU/TU structure. In addition, the CU-synchronized adaptive loop filter (ALF) minimizes the expected average distortion whilst the spatial filtering extreme correction (EXC) reduces the distortion in the specific regions which are important to visual perception. Also, content adaptive dynamic range (CADR) is performed to mitigate rounding effects and to increase the accuracy of intermediate calculation without increasing bit-depth.
- **New intra prediction methods**: to increase the performance of intra coding, four new intra tools are included: arbitrary directional intra (ADI), pixel based template matching (PTM), color component correlation based prediction (CCCP) and multi-parameter intra (MPI). Using these tools, prediction patterns can be provided which cannot be generated efficiently in the conventional way.
- **New entropy coding with explicit scan order signaling**: syntax-based binary arithmetic coder (SBAC) is proposed. In order to increase the efficiency of the entropy coding of transform coefficients, an explicit scan order is signaled from amongst pre-defined scan orders.

Figure 2-1 shows all building blocks to decode the input bit stream. White boxes indicate the newly introduced building blocks. From the following sections, each building block is explained in detail.

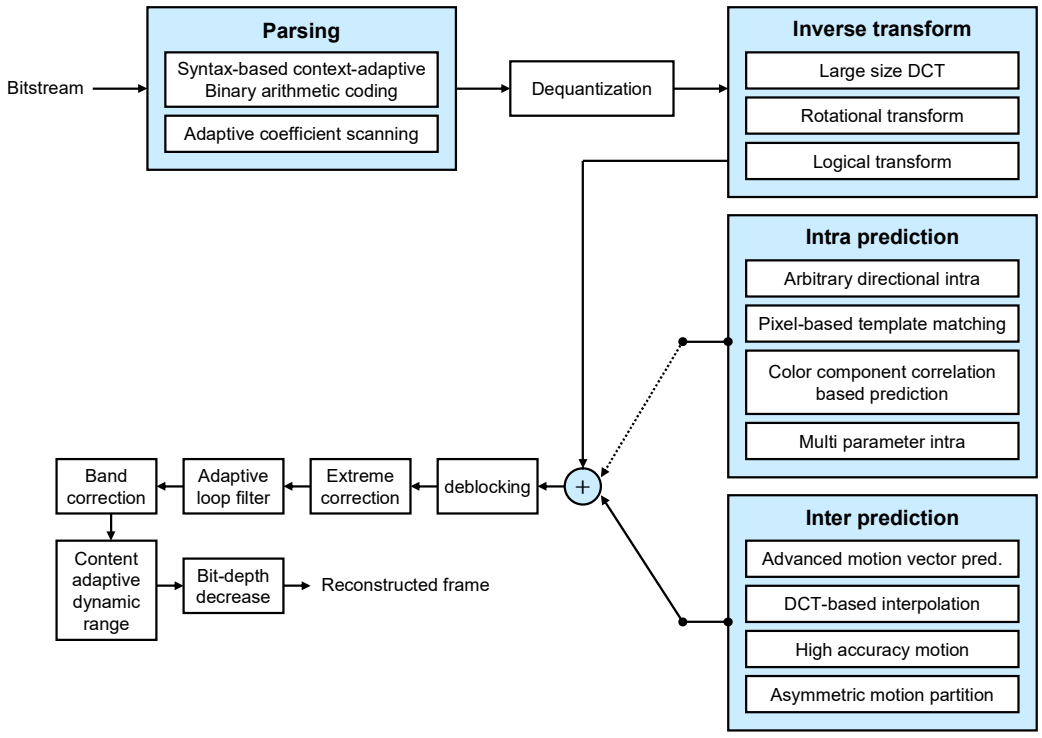


Figure 2-1 All building blocks to decode the input bit stream

2.2 Unit definition

2.2.1 Coding unit (CU)

In this proposal, the coding unit (CU) is defined as a basic unit which has a square shape. Although it has a similar role to the macroblock and sub-macroblock in H.264/AVC, the main difference lies in the fact that CU can have various sizes, with no distinction corresponding to its size. All processing except frame-based loop filtering is performed on a CU basis, including intra/inter prediction, transform, quantization and entropy coding. Two special terms are defined: the largest coding unit (LCU) and the smallest coding unit (SCU). For convenient implementation, LCU size and SCU size are limited to values which are a power of 2 and which are greater than or equal to 8.

It is assumed that a picture consists of non-overlapped LCUs. Since the CU is restricted to be a square shape, the CU structure within a LCU can be expressed in a recursive tree representation adapted to the picture. That is, CU is characterized by LCU size and the hierarchical depth in the LCU that the CU belongs to.

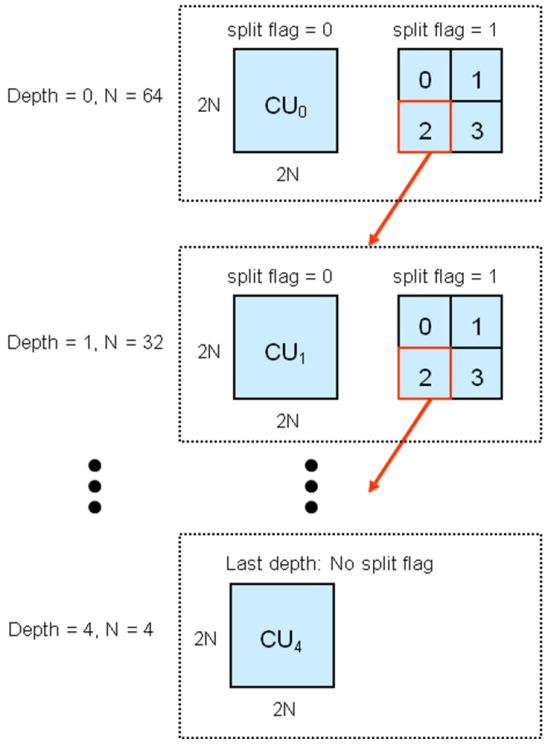


Figure 2-2 Illustration of recursive CU structure (LCU size = 128, maximum hierarchical depth = 5)

Figure 2-2 shows an example where LCU size is 128 and the maximum hierarchical depth is 5. The recursive structure is represented by a series of split flags. For CU_d, which has depth d and size 2Nx2N, the coding of CU is performed in the current depth when split flag is set to zero. When the split flag is set to 1, CU_d is split into 4 independent CU_{d+1} which have depth (d+1) and size NxN. In this case, CU_{d+1} is called a sub-CU of CU_d similar to a sub-macroblock in H.264/AVC. Unless the depth of sub-CU (d+1) is equal to the maximum allowed depth (4 in this case), each CU_{d+1} is processed in a recursive manner. If the depth of sub-CU (d+1) is equal to the maximum allowed depth, further splitting is not allowed. Note that a CU can be further split into PUs, however (see section 2.1.2).

The sizes of LCU and SCU are specified in the Sequence Parameter Set (SPS). The embedded information in the SPS is LCU size (s) and the maximum hierarchical depth (h) in a LCU. For example, if s = 128 and h = 5, then 5 kinds of CU sizes are possible: 128x128 (LCU), 64x64, 32x32, 16x16 and 8x8 (SCU). If s = 16 and h = 2, then 16x16 (LCU) and 8x8 (SCU) are possible; this is a similar block structure to H.264/AVC. Therefore, if the LCU size and maximum hierarchical depth are given, this defines the possible CU sizes which are allowed.

This kind of arbitrary unit representation provides several major benefits. The first benefit comes from the support of CU sizes greater than the conventional 16x16 macroblock. When the region of interest is homogeneous, a large CU can represent the region in a smaller number of symbols than is possible in the case of several small blocks. The evidence for this conclusion has already been presented by several papers both inside MPEG [1][2] and at external conferences [3][4]. Similar approaches have also been followed by recent proposals in MPEG and VCEG [5][6][7].

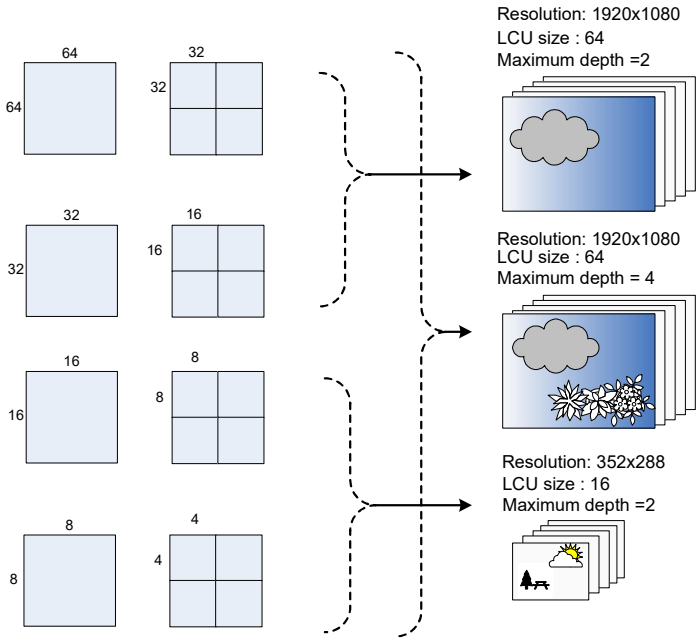


Figure 2-3 Example of LCU size and maximum depth combinations for various resolutions

Furthermore, supporting arbitrary LCU sizes enables the codec to be readily optimized for various content, applications and devices. Compared to the use of fixed size macroblock, support of various LCU sizes is one of the major properties of this proposal. It is especially useful for low resolution video services, which is still commonly used in the market. By choosing LCU size and maximum hierarchical depth appropriately, the hierarchical block structure can be optimized in a better way for the targeted application. In the later stage of the standardization, the range of LCU sizes could be specified in the Profiles and Levels section to match the requirements more specifically. Figure 2-3 shows examples of CU size combinations for several resolutions.

Finally, by eliminating the distinction between macroblock and sub-macroblock and using only one unit type, CU, the multi-level hierarchical structure can be specified in a very simple and elegant way: LCU size, maximum hierarchical depth and a series of split flags. Together with the proposed size-independent syntax representation, it's sufficient to specify syntax items of one general size for the remaining coding tools. This kind of consistency can greatly simplify the specification effort as well as the actual parsing process.

2.2.2 Prediction unit (PU)

Once the splitting process is done, prediction methods are specified for every CU which is not further split i.e. the leaf nodes of the CU hierarchical tree.

Coupled with CU, this proposal introduces a basic unit for the prediction mode: the prediction unit (PU). It should be noted that the PU is defined only for the last-depth CU and its size is limited to that of the CU.

Similar to conventional standards, we define two different terms to specify the prediction method: the prediction type and the PU splitting. The prediction type is one of the values among skip, intra or inter, which roughly describe the nature of the prediction method. After that, possible PU splittings are defined according to the prediction type.

Figure 2-4 shows possible PU splittings according to the different prediction types for a CU of size 2Nx2N. The PU for intra has 2 different possible splittings: 2Nx2N (i.e. no split) and NxN (quarter split). The PU for inter has 8 different possible splittings: 4 symmetric splittings (2Nx2N, 2NxN, Nx2N, NxN) and 4 asymmetric splittings (2NxnU, 2NxnD, nLx2N and nRx2N). A skipped PU can only be 2Nx2N i.e. the whole CU is skipped. The number N is derived from the size of the CU which the PU belongs to. For example, if the size of CU is 128x128, then both 128x128 and 64x64 PUs for intra are possible. For inter prediction, 128x128, 128x64, 64x128, 64x64, 128x32 (for 2NxnU and 2NxnD), 128x96 (for 2NxnU and 2NxnD), 32x128 (for nRx2N and nLx2N) and 96x128 (for nRx2N and nLx2N) are possible.

All information related to prediction is signaled on a PU basis, for instance, the direction of intra prediction for intra or the motion vector difference and reference index for inter. Motion vector prediction and motion compensation are also performed on a PU basis.

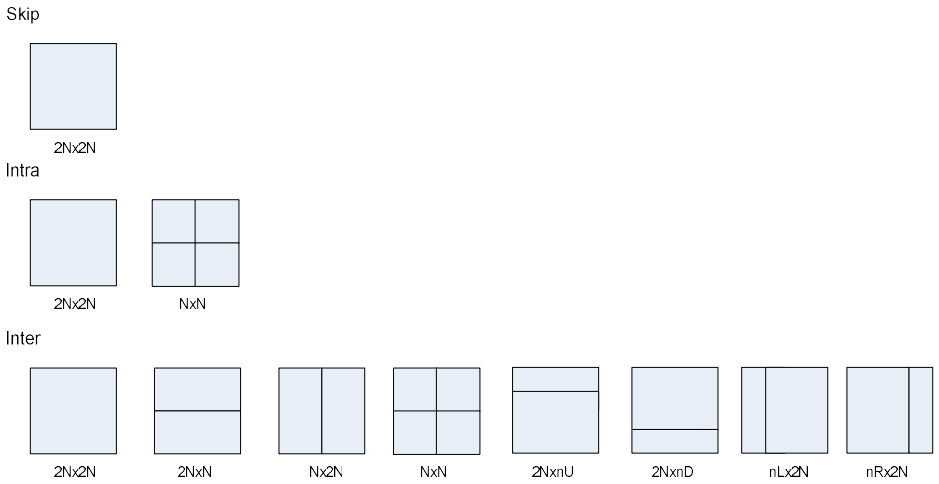


Figure 2-4 Example of possible PU splittings for skip, intra and inter prediction

2.2.3 Transform unit (TU)

In addition to the CU and PU definitions, the transform unit (TU) for transform and quantization is defined separately. It should be noted that the size of the TU may be larger than that of the PU, which is different from previous video standards, but it may not exceed the CU size. However, the TU size is not arbitrary and once a PU structure has been defined for a CU, only two TU partitions are possible. As a result, the size of TU in the CU is determined by the transform unit size flag. If the transform unit size flag is set to 0, the size of TU is the same as that of the CU which the TU belongs to. Otherwise, the TU size is set as NxN or N/2xN/2 according to the PU splitting. Figure 2-5 shows an example of TU when size of CU is 2Nx2N. According to transform unit size flag and PU splitting, the size of TU is specified to be 2Nx2N, NxN or N/2xN/2, as illustrated in the figure below. It should be noted that the case of zero transform unit size flag is designed for performing a transform of CU size regardless of PU splitting. This approach yields coding gains especially for coding semi-random residuals generated by complex texture areas.

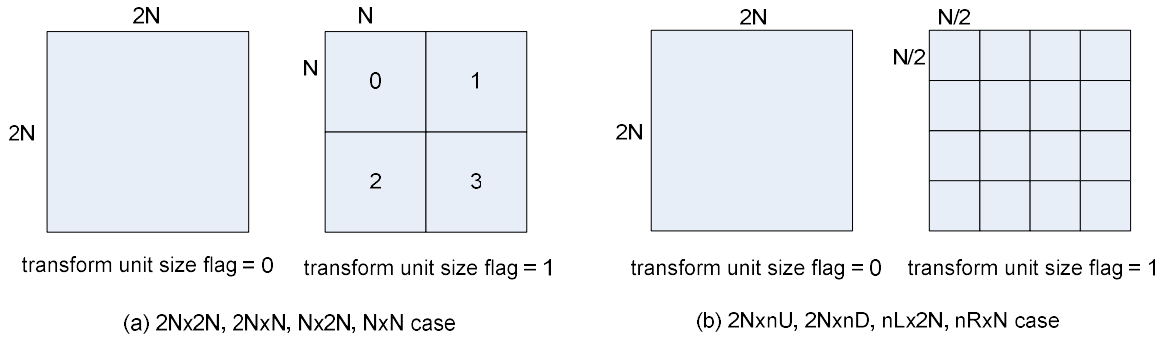


Figure 2-5 Example of transform unit (TU) according to the transform unit size flag

2.2.4 Relationship between CU, PU and TU

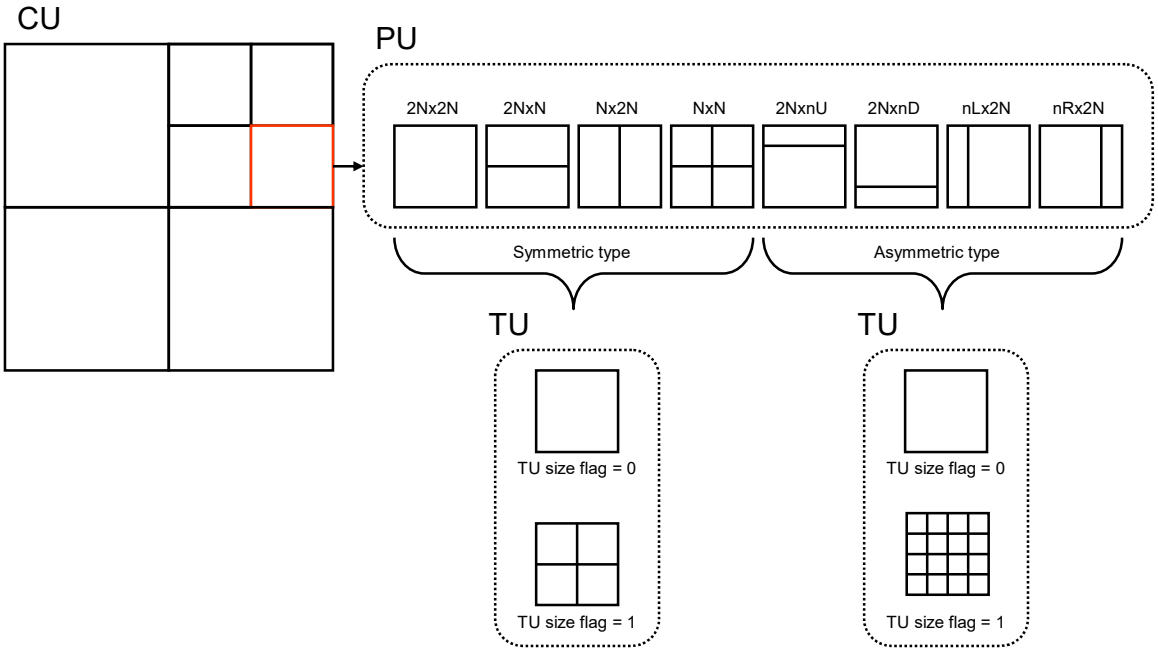


Figure 2-6 Relationship between CU, PU and TU

Figure 2-6 shows the relationship between the three different unit definitions, CU, PU and TU. For any size of CU given, a series of split flags is used to specify smaller size CUs inside the CU. For every CU which does not split any more, a PU is defined specifying how the prediction can be generated. Note that the PU need not be a square shape and virtually any kind of new PU splitting could be added without changing the overall structure. TU size is specified by TU size flag and PU splitting. If the TU size flag is zero, TU size is always equal to CU size whereas TU size is defined differently depending upon PU splitting when the TU size flag is one. In this latter case, the TU size is chosen so that the transform blocks do not cross PU block boundaries. Table 2-1 summarizes the possible combinations of CU, PU, and TU based on the value of split flag, prediction type, PU splitting and TU size.

Table 2-1 Possible combinations of CU, PU and TU

<table>
<tr><td colspan="4">split flag = 0 (CU_d processing, current depth = d)</td><td>split flag = 1 (NxN split)</td></tr>
<tr><td>Prediction type</td><td colspan="2">PU splitting</td><td colspan="2">TU size</td><td rowspan="3">Recursive CU processing for CU_{d+1} whose size is NxN</td></tr>
<tr><td>INTRA (only for 2Nx2N, NxN)</td><td>symmetric type</td><td>asymmetric type</td><td>TU size flag = 0</td><td>TU size flag = 1</td></tr>
<tr><td>INTER (for all PU splitting)

SKIP (only for 2Nx2N)</td><td>2Nx2N
2NxN
Nx2N
NxN</td><td>2NxnU
2NxnD
nLx2N
nRx2N</td><td>2Nx2N</td><td>NxN (symmetric type)
N/2xN/2 (asymmetric type)</td></tr>
</table>

2.2.5 Size-independent syntax representation

In the proposed codec, the maximum depth for CU hierarchy is arbitrary and potentially larger than the maximum depth allowed in the H.264/AVC block hierarchy. Therefore, the proposal includes a size-independent syntax representation which specifies all syntax elements in a consistent way independent of the CU size. By contrast, in H.264/AVC block level syntax elements such as transform_8x8_mode_flag, coded_block_flag and intra_pred_modes are coded differently depending on whether the block size is 16, 8 or 4. Table 2-2 shows the recursive part of the CU syntax in tabular form.

Table 2-2 Recursive CU syntax in tabular form

read_CU (currDepth) {	Descriptor
split_flag	ae(v)
if (split_flag && currDepth < maxAllowedDepth) {	
for (cuIdx = 0; cuIdx < 4; cuIdx++)	
read_CU (currDepth+1)	
}	
else {	
CU_data (currDepth) /* read syntax elements for arbitrary size CU */	
}	
}	

As shown in the table, the splitting process for the CU can be specified recursively and all other syntax elements for the leaf CU are defined in the same way independent on the CU size. This kind of representation is very useful in terms of reduced parsing complexity and improved clarity if a large hierarchical depth is allowed.

2.3 Motion representation

In this proposal, motion fields are represented by the motion vectors obtained from each PU. As specified in Section 2.2.2, four kinds of asymmetric shape of motion partition are supported to enhance the accuracy of the motion field representation. For the motion accuracy, quarter-pel precision with 1/12 optional refinement is supported. Motion vector prediction is applied only for the quarter-pel accuracy level and multiple motion vector predictor candidates are considered to improve the prediction efficiency. We also propose the use of non-cascaded spatial filtering for every motion accuracy position to minimize the motion compensation complexity when higher motion vector accuracy is used whereas the previous standards typically use cascaded spatial filtering, such as well-known 6-tap with bi-linear interpolation for the quarter-pel motion accuracy.

2.3.1 Asymmetric motion partition (AMP)

The hierarchical block structure of the proposed video codec makes the representation of image patterns more accurate and efficient. To further improve the efficiency, asymmetric motion partition (AMP) is introduced by allowing asymmetric partitions of a block for inter prediction. Figure 2-7 shows an example of the AMP where a 64x64 block is asymmetrically partitioned into 64x16, 64x48, 16x64, or 48x16 blocks, instead of two 64x32 or 32x64 blocks. Note that the motion partitions shown in the figure correspond to 2NxnU, 2NxnD, nLx2N, and nRx2N, respectively. AMP improves the coding efficiency, since irregular image patterns, which otherwise would be constrained to being represented by a symmetric partition, can now be more efficiently represented without requiring further splitting.

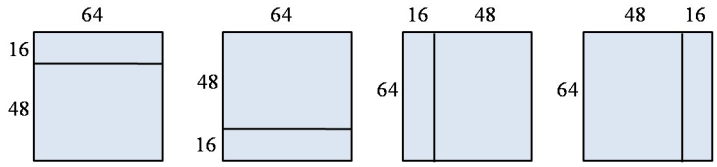


Figure 2-7 Example of asymmetric motion partitions for 64x64 block

2.3.2 Advanced motion vector prediction (AMVP)

The bit rate for encoding motion information is reduced by utilizing predictive coding of motion vectors. Conventional video coding standards use a simple shared rule between encoder and decoder. For example, H.264/AVC predicts its motion vector using a spatial median, based on neighboring blocks. The median is adapted depending on the availability of these neighboring blocks. A few competition-based schemes have also been proposed, which usually select the best predictor from a given set, and send the index of this predictor as side information. The set is typically composed of three neighboring motion vectors and a temporal motion vector, where these predictors are exhaustively compared [8]. However, such a competition-based scheme needs to consider more available neighboring motion vectors and temporal motion vectors when dealing with large block representation and a flexible structure.

In this proposal, an advanced motion vector prediction method is introduced which is adapted to large block representation and a flexible temporal structure. It allows the selection of the best predictor from a given set, where the set is composed of three spatial motion vectors, a median motion vector and a temporal motion vector.

In the predictor set, the three spatial motion vectors are chosen from among those above, to the left and from each applicable corner. In each direction, the first available motion vector is inspected among neighboring blocks and it is chosen as a spatial motion predictor once it is found. The corner blocks include the right top, the left top as well as the left below neighboring block depending on the current block position.

The temporal motion predictor is given by the nearest reference frame and is scaled according to temporal distance. For B pictures, the direction to get the reference frame is indicated for each slice, to provide the better temporal motion predictor among forward and backward reference frames.

The ordering of candidates in the set is important, to reduce the overhead of signaling the position of the best motion predictor in the predictor set. The ordering of the set is adapted depending on the current prediction mode to place the most probable motion predictor in the first position, since minimum overhead occurs if the first candidate is chosen as the best predictor.

The overhead of signaling overhead the index of the best predictor can be reduced further by minimizing the number of candidates in the set. Duplicated motion vectors are simply removed from the set and motion predictors which could not have been used in the encoder are also removed after possible candidates are determined from the transmitted MVD value [11]. If only one motion predictor is available, the best motion predictor is implied by the motion predictor without overhead.

2.3.3 Improved skip and direct mode

As the video resolution and the size of coding units increase, the provision of intermediate types of mode, between the simplest and the most complex mode, becomes useful to improve coding efficiency. In H.264/AVC, the simplest modes like SKIP or DIRECT mode send no motion information whatsoever whilst the other modes require both motion and texture information.

In this proposal, new skip and direct concepts are presented. Regardless of slice type, both skip and direct modes are supported; the difference between skip and direct is whether texture information is sent or not. The motion of skip and direct is inferred by the best motion vector predictor signaled by its position in the motion predictor set. Furthermore, uni-directional direct modes in B slices are supported in addition to the bi-directional direct mode.

2.3.4 DCT-based interpolation filter (DIF)

Motion compensation is the key factor for efficient video compression. Compensation for motion with fractional pel accuracy requires interpolation of the reference signal. A conventional approach uses a 6-tap Wiener filter for half-pel generation. For higher accuracy (for example 1/4 pel), first a Wiener filter is applied and then a bi-linear combination of integer and half pel values is used to provide 1/4 pel accuracy interpolation.

This proposal replaces the combination of Wiener and bilinear filters by directly providing the interpolated value at the desired fractional accuracy. More specifically, only one filter operation is needed to provide interpolated pixel to any pixel accuracy, thus motion compensation process can be simplified in the implementation point of view and complexity can be reduced for quarter-pel accuracy when a combination of 6-tap and bilinear filter is used. Furthermore, this tool supports the unified way to generate the interpolation filters supporting higher than 1/4 pel motion vector accuracy, e.g. 1/8 pel or even higher. More detailed explanation supporting higher motion vector accuracy than quarter-pel is provided in Section 0.

2.3.4.1 DCT basis decomposition

This section explains the motivation and conceptual realization of the new interpolation. Assume that the integer points values $\{p_i\}$, $i=-(M-1),\ldots,M$ are known and that the value p in position $0 \le \alpha \le 1$ is to be predicted.



Figure 2-8 Known integer pel values (shadowed) and predicted fractional pel value p.

Assume that the values $\{p_i\}$, $i=-(M-1),\ldots,M$ are integer points values of a smooth function $p(x)$.
The forward DCT provides the transform coefficients set:

$$C_k = \frac{1}{M}\sum_{l=-M+1}^{M} p(l)\cos\left(\frac{(2l-1+2M)k\pi}{4M}\right). \quad (1)$$

The inverse DCT

$$p(x) = \frac{C_0}{2} + \sum_{k=1}^{2M-1} C_k \cos\left(\frac{\pi(2x-1+2M)k}{4M}\right) \quad (2)$$

returns absolutely exactly *p(x)* for integer *x=-(M-1),-(M-2),…,M*.
A fractional point value can be calculated by using the corresponding shift as the basis function argument.

$$p(\alpha) = \frac{C_0}{2} + \sum_{k=1}^{2M-1} C_k \cos\left(\frac{\pi(2\alpha-1+2M)k}{4M}\right). \quad (3)$$

The sequential processing of forward DCT decomposition (2) and inverse DCT with a shifted argument for the basis functions (3) therefore leads to a new interpolation realization.

However, the actual usage of a forward-inverse transform pair is not desirable because of its high computational complexity. In this proposal, transform operation pairs and intermediate phase shift were merged into a single spatial domain filter.

2.3.4.2 Example of filter coefficients

The computational complexity of filtering depends on the number of filter taps: 2M. For M=3 computational complexity is comparable to a 6-tap Wiener filter. A larger number of filter taps provides better interpolation quality. Several pre-calculated filter coefficients for each fractional position are given in the Table 2-3 and Table 2-4 for 6 tap and 12 tap, respectively. The filter coefficients are scaled by 256. A higher bit depth of filter coefficients leads to more accurate interpolation; for comparison, H.264/AVC uses 5 bits for filter coefficients.

Table 2-3 Filter coefficients of 6-tap DIF filter

α	2M = 6 (6-tap filter)
-1/12	{ -4, 19, 254, -19, 8, -2,}
1/12	{ 4, -16, 252, 22, -8, 2,}
1/6	{ 6, -28, 242, 48, -17, 5,}
1/4	{ 9, -37, 227, 75, -25, 7,}
2/6	{ 11, -42, 208, 103, -33, 9,}
5/12	{ 12, -44, 184, 132, -39, 11,}
1/2	{ 11, -43, 160, 160, -43, 11,},
7/12	{ 11, -39, 132, 184, -44, 12,}
2/3	{ 9, -33, 103, 208, -42, 11,}
3/4	{ 7, -25, 75, 227, -37, 9,},
5/6	{ 5, -17, 48, 242, -28, 6,}

Table 2-4 Filter coefficients for 12-tap DIF filter

α	2M = 12 (12-tap filter)
-1/12	{ 1,-3, 5,-10, 22,253,-19, 10, -6, 4,-2, 1 }
1/12	{-1, 3, -5, 9,-19,253, 23,-10, 6, -4, 2,-1 }
1/6	{-2, 5, -9, 16,-34,244, 49,-21, 12, -7, 4,-1 }
1/4	{-1, 6,-12, 21,-43,229, 75,-30, 17,-10, 5,-1 }
2/6	{-3, 8,-15, 26,-50,211,105,-40, 22,-13, 7,-2 }
5/12	{-3, 9,-16, 28,-53,188,134,-47, 26,-15, 8,-3 }
1/2	{-2, 7,-15, 28,-52,162,162,-52, 28,-15, 7,-2 },
7/12	{-3, 8,-15, 26,-47,134,188,-53, 28,-16, 9,-3 }
2/3	{-2, 7,-13, 22,-40,105,211,-50, 26,-15, 8,-3 }
3/4	{-1, 5,-10, 17,-30, 75,229,-43, 21,-12, 6,-1 },
5/6	{-1, 4, -7, 12,-21, 49,244,-34, 16, -9, 5,-2 }

2.3.4.3 Low complexity optimization

An efficient hardware implementation requires multiplication-free representation for all calculations. In order to minimize the number of arithmetical operations (shifts and additions), the filter coefficients in Table 2-3 and Table 2-4 can be slightly changed without significant change of performance. Table 2-5 and Table 2-6 contain optimized filter coefficients together with the required number of arithmetical operations. In the case of 6 tap filters half-pel interpolation, the DIF optimized coefficients are the same as the Wiener filter coefficients in H.264/AVC multiplied by 8. Hence, the computational complexity is also the same as for the half-pel interpolation filter in H.264/AVC. With the same computational complexity, the 6-tap DIF outperforms the Wiener + bi linear filter combination in H.264/AVC. The 12-tap DIF outperforms the Adaptive Interpolation Filter (AIF) without requiring any content adaptation.

Table 2-5 Optimized Filter coefficients for 6-tap DIF filter

α	2M = 6 (6-tap filter)
1/4	{ 8, -32, 224, 72, -24, 8 } (8 additions, 4 shifts)
1/2	{ 8, -40, 160, 160, -40, 8 } (6 additions, 3 shifts)
3/4	{ 8, -24, 72, 224, -32, 8 } (8 additions, 4 shifts)

Table 2-6 Optimized Filter coefficients for 12-tap DIF filter

α	2M = 12 (12-tap filter)
1/4	{-1, 5, -12, 20,-40, 229, 76, -32, 16, -8, 4,-1 } (18 additions, 6 shifts)
1/2	{-1, 8, -16, 24,-48, 161, 161, -48, 24, -16, 8,-1 } (15 additions, 4 shifts)
3/4	{-1, 4, -8, 16, -32, 76, 229, -40, 20, -12, 5,-1 } (18 additions, 6 shifts)

2.3.5 High accuracy motion (HAM)

Generally speaking, if the motion accuracy for motion compensation is increased, smaller prediction error can be obtained. However, more accurate motion representation also generally leads to an increase in the bit-rate required to convey the motion information. The additional overhead mainly comes from two facts. Firstly, when the motion accuracy is increased the magnitude of motion vector is also increased. Secondly, motion vector prediction is more difficult since the range of motion vector values is increased. As a result, 1/8 pel motion accuracy (i.e. twice the 1/4 pel accuracy in H.264/AVC) may result in a reduction in the overall compression efficiency, except for particular sequences [9].

In this proposal, efficient representation is presented to utilize higher motion accuracy than 1/4 or 1/8 pel without any negative effects. Additional motion information beyond 1/4 pel accuracy is represented by simple refinement information instead of requiring a more accurate motion vector. More specifically, high accuracy motion is represented by MV (with lower motion accuracy, for example 1/4 pel) + refinement. The benefit of this representation is as follows. Firstly, the magnitude of the motion vector is not increased because the accuracy of motion vector is not changed, only the refinement. Secondly, since motion vector prediction is done without refinement information, there is no increase in the difficulty of motion vector prediction. Finally, the overhead is minimized since the refinement information is sent only when it is useful. For example, for each motion vector component, the first bit of refinement indicates whether the refinement is needed or not. If the bit is zero, it means that there is no more refinement information and next bit is not needed. If the bit is not zero, following bit indicates -1/12 or +1/12 refinement.

Although the refinement information can represent any accuracy, the current design uses 1/12 instead of 1/8 pel to give a more compact representation. If 1/8 pel accuracy is used instead of 1/12, there may be up to four different ways of representing the same position.

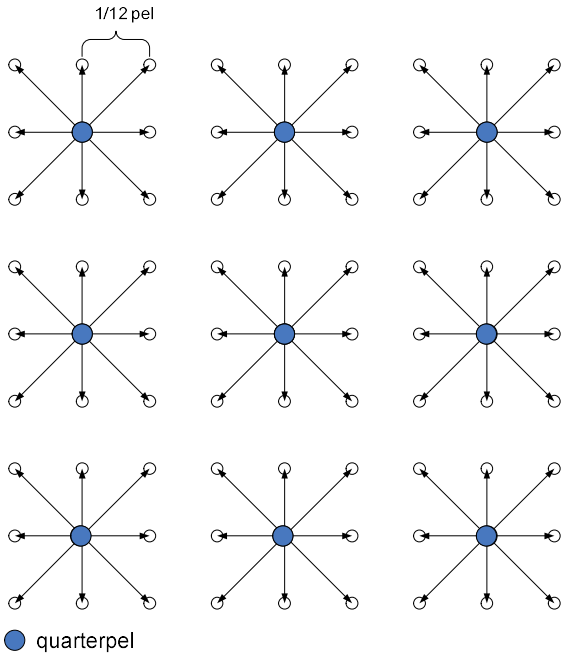


Figure 2-9 High accuracy motion grid

2.4 Intra-frame prediction

In this proposal, four intra tools are integrated to generate the final intra prediction results. The arbitrary directional intra (ADI) is the main intra prediction tool; it is an extension of the directional intra tool in H.264/AVC. The pixel based template matching (PTM) provides different prediction patterns which cannot be generated from directional pattern extrapolation or by simple calculation from neighboring pixels. The color component correlation based prediction (CCCP) tool improves the coding efficiency of chroma samples by exploiting the correlation between luma and chroma samples. The multi parameter intra (MPI) tool smoothes the luma predicted patterns while maintaining the meaningful details, which results in better coding efficiency from the transform. The MPI also provides variations to predicted patterns from ADI and PTM. Figure 2-10 shows an example of various target regions related to each intra prediction tool.

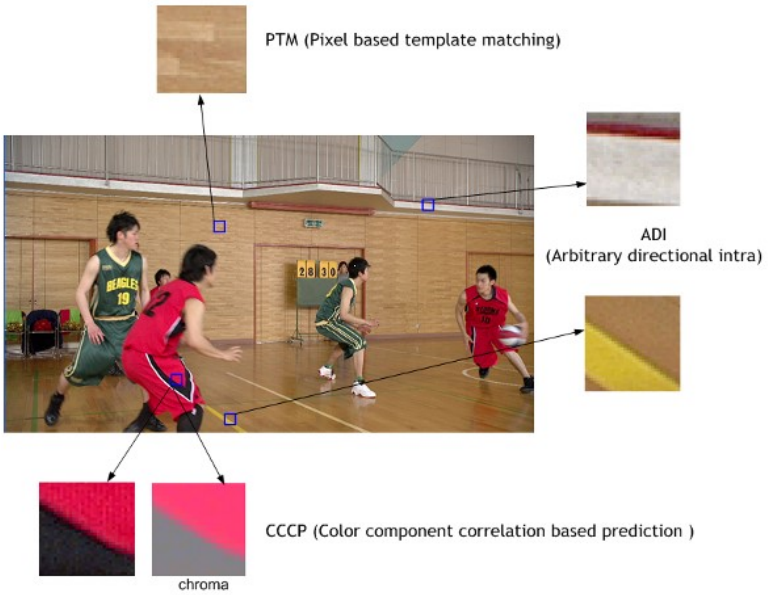


Figure 2-10 An example of target areas for various intra prediction tools

2.4.1 Arbitrary directional intra (ADI)

The proposed codec structure supports various sizes of coding units and transforms. This new codec structure provides large blocks and large transforms which are known to improve coding efficiency for HD resolutions. But if prediction methods are not designed accordingly, the benefits from the new codec structure cannot be maximized. Arbitrary directional intra (ADI) is designed to fit into this flexible codec structure.

In H.264/AVC intra coding, only 4 intra prediction modes are available for 16x16 blocks, since 16x16 blocks in lower resolution video generally represent either homogenous regions which can be described by the 4 prediction modes (vertical, horizontal, DC, plane) or else overly complicated patterns which cannot be predicted by a single prediction. But in HD sequences, even relatively large blocks (>=16x16) cover only a small part of objects or backgrounds, which can be described as homogeneous texture patterns with variations such as gradation or directional pattern extrapolation. Therefore additional performance improvement can be expected by applying directional prediction for these large blocks.

However, applying the same 9 directional predictions from conventional 4x4 intra prediction does not provide enough variation to describe the likely patterns in large blocks in HD sequences. Therefore, more prediction modes (up to 33) are provided in the ADI. Different block sizes are used for different image patterns for intra prediction; the number of available prediction modes for each block is adjusted according to the block size.

As in H.264/AVC intra prediction, the ADI generates prediction pixels by directional extrapolation or calculation using the nearest boundary pixels of the already decoded area. But in ADI, even boundary pixels from the left down region may be used as context pixels for prediction as depicted in Figure 2-11.

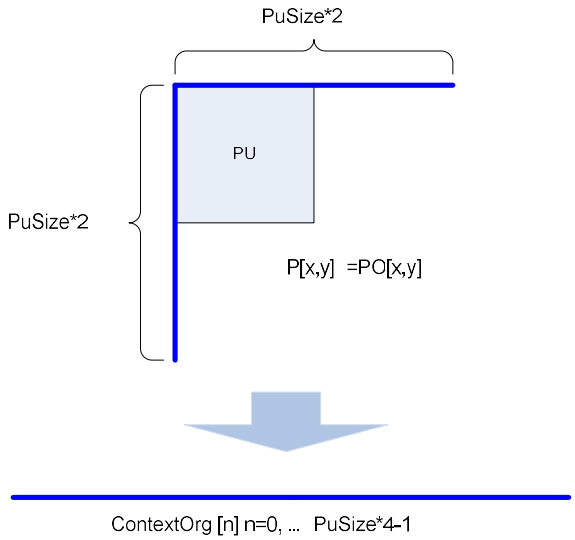


Figure 2-11 Example of context pixels for ADI

Whilst the 9 prediction modes are defined separately as Vertical, Horizontal, DC, Diagonal Down-Left, Diagonal Down-Right, Vertical-Right, Horizontal-Down, Vertical-Left, and Horizontal-Up in H.264/AVC, most of prediction modes in ADI are defined by integer pair information (dx, dy). The (dx, dy) pair represents the direction which each mode uses for context pixel extrapolation, as illustrated in Figure 2-12 below.

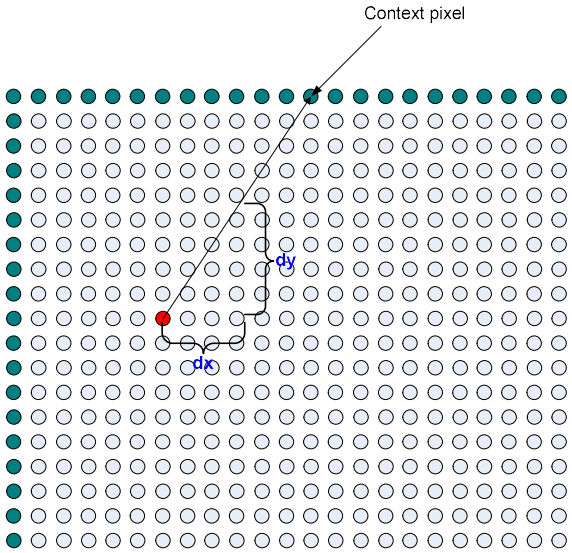


Figure 2-12 prediction methods defined by (dx, dy)

The available prediction modes for ADI are as follows.

Table 2-7 Available prediction modes for ADI

Mode description	Pred mode number
Vertical	0
Horizontal	1
DC	2
Plane	3
(dx_i, dx_y)	4 ~ 31
Bi-linear	32

The prediction modes used for each coding unit sizes can be found in the following table.

Table 2-8 Prediction modes used for each coding unit sizes

Prediction Unit size	Number of prediction modes	Used prediction modes
4x4	9	0~2, 4~9
8x8	9	0~2, 4~9
16x16	33	0~32
32x32	33	0~32
64x64	5	0~5
128x128	5	0~5

Depending on the prediction unit size, the number of filtering operations that may be applied to the context pixels varies, which provides different prediction patterns.

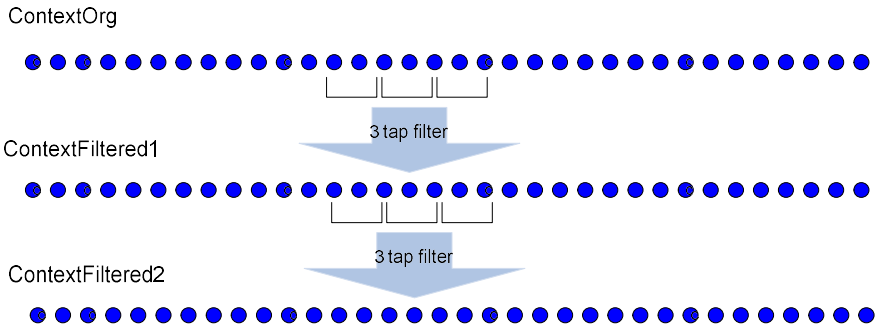


Figure 2-13 Example of context pixel filtering

For the chroma component, different prediction modes are applied for each prediction size. The prediction modes are chosen from 0: Vertical, 1: Horizontal, 2: CCCP mode, 3: Plane and 4: PTM mode.

2.4.2 Multi-parameter intra (MPI)

Directional intra prediction often results in non-smooth and unnatural patterns which are not transform-friendly. It is possible to reduce this problem by using a smoothing filter, but conventional smoothing filters tend to also remove meaningful details from the predicted results.

The purpose of MPI is to generate more natural patterns by applying different post processing filters to the prediction results from directional intra. Whilst conventional smoothing filters use NxN symmetric filters, MPI uses a 4 point filter for each pixel which maintains the directional patterns even after the smoothing. Figure 2-14 depicts the prediction signal after ADI and MPI is applied.

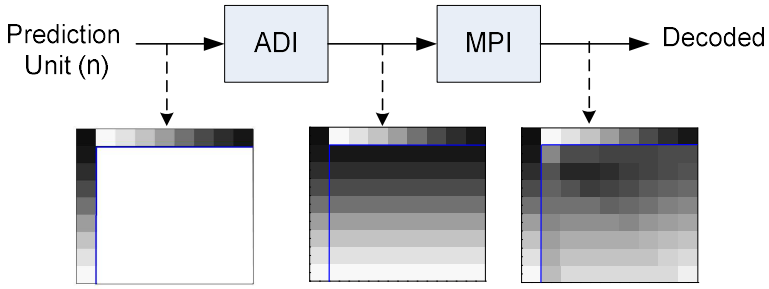


Figure 2-14 Combination of ADI and MPI

For each prediction unit, MPI post-processing is applied after the ADI predictions. If MPI_index=0, no MPI processing is applied. If MPI_index = 1, then a 4 point based filter is applied as in the following equation and the figure (Figure 2-15) where pred' represents the predicted results after MPI and pred defines the predicted results from ADI or already reconstructed pixels.

pred' [x,y]=(pred[x,y]+pred[x-1,y]+ pred[x,y-1]+pred[x,y+1]+2)>>2.

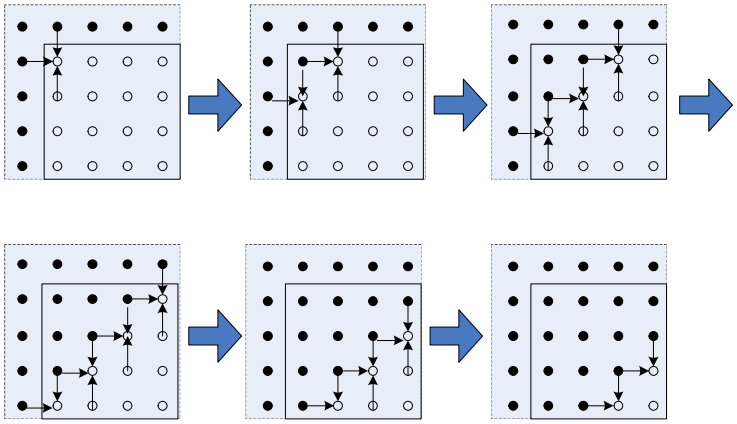


Figure 2-15 MPI process description

2.4.3 Color Component correlation based prediction (CCCP)

In most video codecs, chroma sample prediction is performed after the luma sample reconstruction. Therefore, any information inferred from reconstructed luma samples can be used for chroma sample prediction. The CCCP exploits the information of region partitions analyzed from the reconstructed luma samples to improve the coding efficiency of chroma samples.

In the CCCP, region partitions are estimated by performing segmentation on the reconstructed luma samples. The derived segmentation map is used to predict chroma samples. Based on the estimated segmentation map, different reconstructed samples in the neighbourhoods are used to predict each region.

The procedure of CCCP method is illustrated in Figure 2-16. Firstly, reconstructed luma samples are down-sampled by bi-linear filter to match the chroma block size for 4:2:0 chroma format case. Then, simple thresholding using the mean value method is exploited in order to generate the segmentation map. After that, two mean values for the chroma predictions according to the generated map are calculated and assigned to the inner part of chroma block. Finally, a 3x3 averaging filter is applied to smooth the prediction.

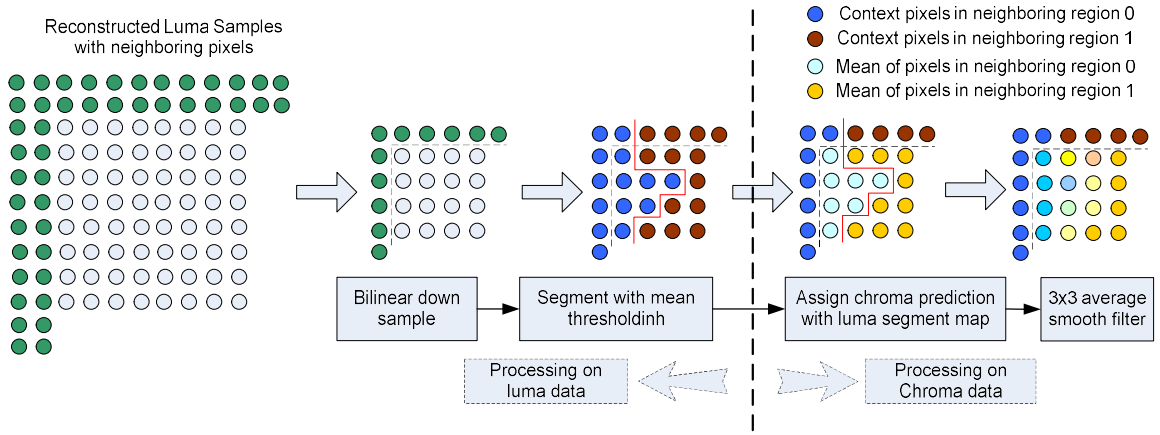
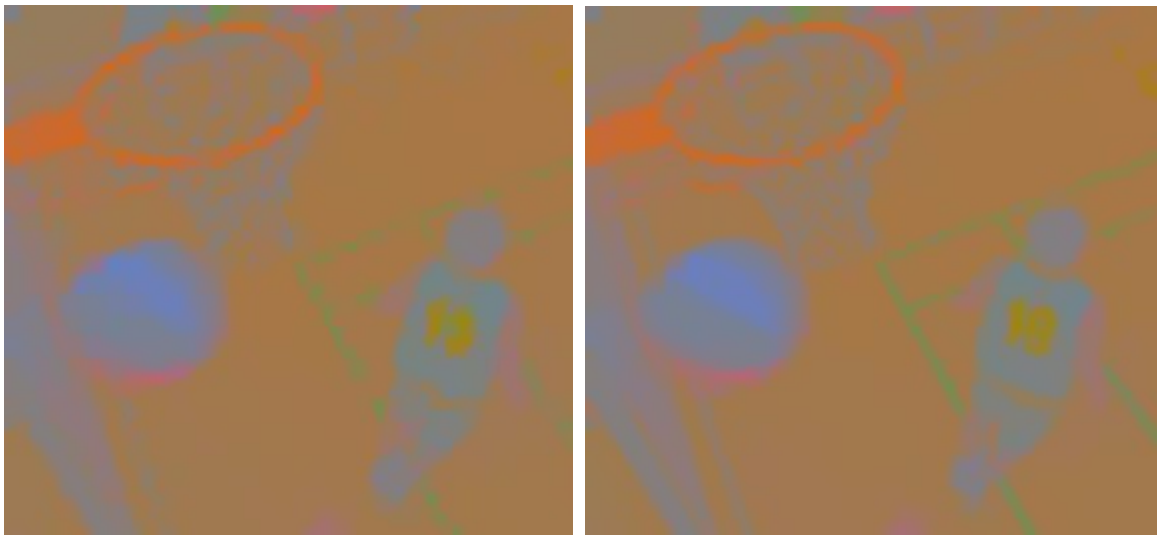


Figure 2-16 CCCP processing flow

Since CCCP utilizes already reconstructed luma blocks, it can generate complex shapes of object, which cannot be predicted well by the directional intra prediction technique. Figure 2-17 shows one typical example from the BasketballPass sequence.



(a) Prediction of directional prediction (b) Prediction of CCCP

Figure 2-17 Visual prediction examples: (a) directional prediction (b) CCCP prediction

2.4.4 Pixel based template matching (PTM)

Whilst directional extrapolation based intra prediction tools are very efficient for regions with linear patterns, they show poor performance in regions with repeated regular patterns. Many previous intra prediction tools used some form of template matching to reduce this problem.

In this proposal, pixel based template matching is employed. Performing template matching on the pixel level results in different effects compared with previous techniques with a large template. A key difference is that already predicted samples in the same prediction unit are also used as template and search candidates for further pixel prediction.

The principle is illustrated in Figure 2-18 below. To predict a sample (PR), three neighbouring pixels (T0, T1 and T2) from above, left, and above-left are used as the template. After searching using this template, the pixel position with minimum template difference is found in the L-shaped search region (reconstructed or already predicted) of the current pixel. The chosen pixel (C) is set to the prediction value of the current pixel. Predicted pixels are also used as the template in the PTM.

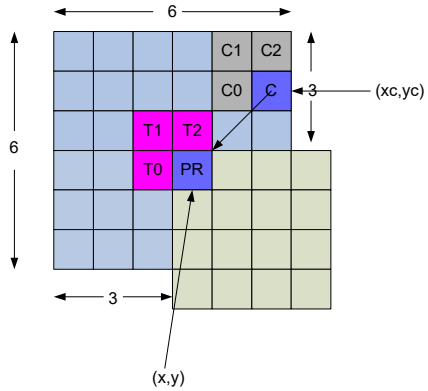


Figure 2-18 Pixel based template matching

2.5 Spatial transforms

2.5.1 Large transform (16x16, 32x32 and 64x64)

For smooth data, a large transform has several advantages such as better energy compaction and reduced quantization error. In HD sequences, most image patterns in a macroblock represent a small part of objects or backgrounds which can be described as homogeneous texture patterns with little variation. Therefore, the coding efficiency of high resolution video can be improved by the use of large transforms as well as large block sizes. By contrast, H.264/AVC supports only 4x4 and 8x8 transform sizes. In this proposal, larger transform sizes can be chosen (or set) for each coding unit (blocks or partitions). Transform sizes larger than the prediction unit can also be supported.

In this proposal, 3 additional sizes of transform are included: 16x16, 32x32 and 64x64. In large transform design, it is very important to minimize complexity. The transforms in this proposal are based on Chen's fast DCT algorithm [10]. Chen's algorithm is not the fastest one, but it has reduced implementation complexity due to the regular butterfly structure. Moreover, it is readily extensible to larger transform sizes.

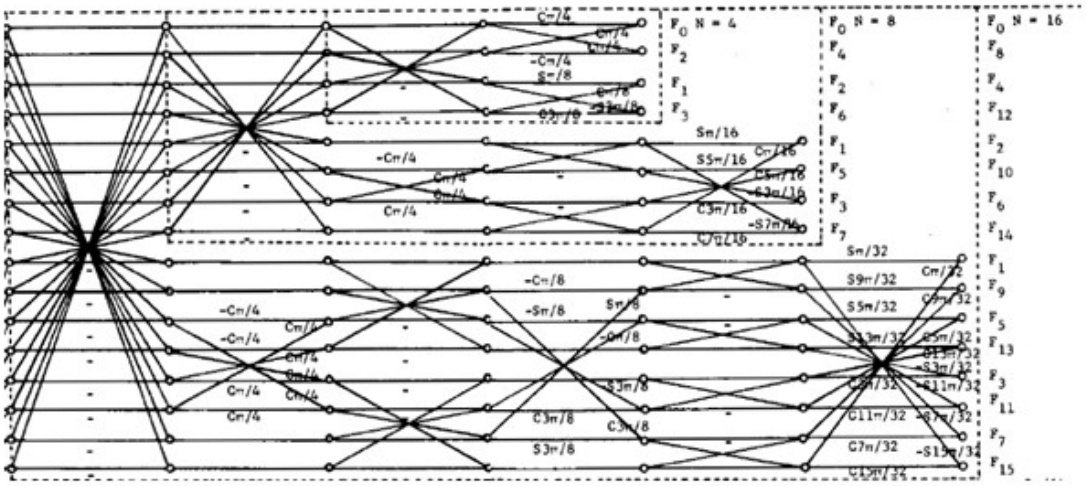


Figure 2-19 Signal flow graph of Chen's fast 16-point DCT transform

Figure 2-19 shows the signal flow graph of Chen's fast factorization of 16 point transform. In this figure, multiplication constants are represented by sinusoidal functions of specific angles, requiring floating point operations. To solve this problem, we scale and approximate the factors by fixed precision using pre-defined values, which can be calculated by cost effective shift operations. The approximated constants are shown in Table 4-8 the pre-defined precision value is 64 in this case. Here, ak's are approximated values of cos(k*pi/32) for k =1,2,…,15.

Table 2-9 Approximated constants for 16 point transform

a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15
63/64	62/64	61/64	59/64	56/64	53/64	49/64	45/64	40/64	35/64	30/64	24/64	18/64	12/64	6/64

When the constant is approximated by the dyadic rational, the transform is no longer truly orthogonal. However, this minor non-orthogonality does not result in any perceptible negative effect on the compression performance, whilst the complexity can be significantly reduced.

2.5.2 Rotational transform (ROT)

DCT is the most widely used transform in block based video and image codecs. The input to the transform is the residual, i.e. the difference between the prediction and the original signal. But it is well-known that the DCT basis functions are not optimal for some types of residual signal. For example, the residual signal having strong diagonal components cannot be represented efficiently with the DCT basis vectors.

Typically, directional transform schemes are exploited to solve this problem since spatial domain rotation makes it dificult to maintain the original square shape of the block and generally requires many floating point operations. However, it's hard to develop directional transform cores, especially since many transforms of different sizes are needed. Instead of introducing completely new transform cores, ROT was designed as a 2nd transform that could be applied after the DCT operation.

The main idea of ROT is to change the coordination system of the transform basis, instead of direct rotation of the input source. For this purpose, the following matrices are defined as

$$R_{vertical}(\alpha_1,\alpha_2,\alpha_3) =$$
$$= \begin{pmatrix} \cos\alpha_1\cos\alpha_3 - \sin\alpha_1\cos\alpha_2\sin\alpha_3 & -\sin\alpha_1\cos\alpha_3 - \cos\alpha_1\cos\alpha_2\sin\alpha_3 & \sin\alpha_2\sin\alpha_3 & 0 \\ \cos\alpha_1\sin\alpha_3 + \sin\alpha_1\cos\alpha_2\cos\alpha_3 & -\sin\alpha_1\sin\alpha_3 + \cos\alpha_1\cos\alpha_2\cos\alpha_3 & -\sin\alpha_2\cos\alpha_3 & 0 \\ \sin\alpha_1\sin\alpha_2 & \cos\alpha_1\sin\alpha_2 & \cos\alpha_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$R_{horizontall}(\alpha_4,\alpha_5,\alpha_6) =$$
$$= \begin{pmatrix} \cos\alpha_4\cos\alpha_6 - \sin\alpha_4\cos\alpha_5\sin\alpha_6 & -\sin\alpha_4\cos\alpha_6 - \cos\alpha_4\cos\alpha_5\sin\alpha_6 & \sin\alpha_5\sin\alpha_6 & 0 \\ \cos\alpha_4\sin\alpha_6 + \sin\alpha_4\cos\alpha_5\cos\alpha_6 & -\sin\alpha_4\sin\alpha_6 + \cos\alpha_4\cos\alpha_5\cos\alpha_6 & -\sin\alpha_5\cos\alpha_6 & 0 \\ \sin\alpha_4\sin\alpha_5 & \cos\alpha_4\sin\alpha_5 & \cos\alpha_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $R_{vertical}$ and $R_{horizontal}$ represent the rotation matrices for horizontal and vertical directions with the rotation angles of (α1, α2, α3) and (α4, α5, α6), respectively. In this proposal, only four out of all possible rotation angles are quantized and used, in order to minimize the encoder-side complexity. Furthermore, since ROT is a second transform, not all of the DCT coefficients are processed. As shown in Figure 2-20, for the TUs of sizes larger than 4x4, only 8x8 low-frequency areas are rotated by multiplying by the ROT matrix since most coefficients are already compacted into these low-frequency areas. The ROT process is applied to all transform sizes but only for intra predicted residuals.

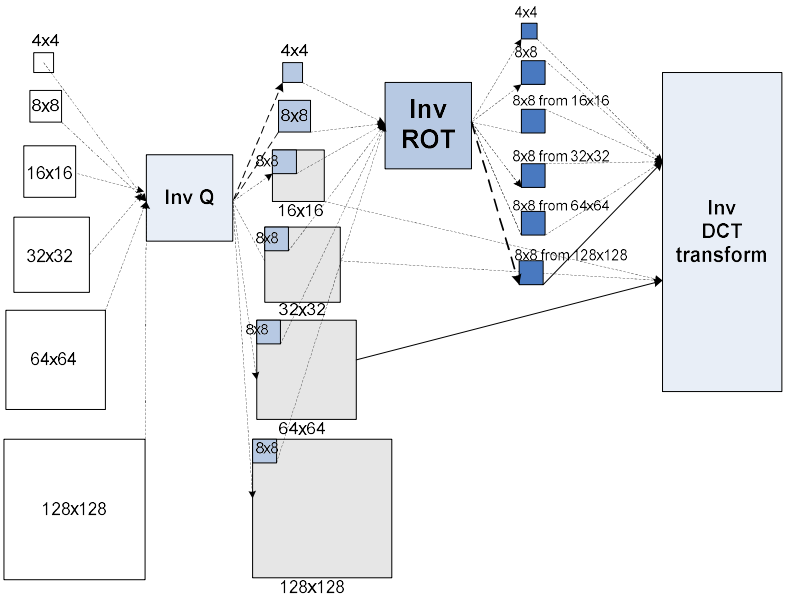


Figure 2-20 Overview of ROT process

2.5.3 Logical transform (LOT)

The LOT allows the input residual size to be bigger than the maximum physical transform size without multiple uses of transforms. If the maximum physical transform size is NxN and the residual to be transformed is of size 2Nx2N, the LOT process is applied to the residual. First, a 2Nx2N 5-3 integer wavelet transform is performed and for the LL-band signal, then a normal DCT is applied. The LOT process is roughly equivalent to taking the NxN low-frequency components of 2Nx2N DCT, thus it works well for the smooth regions which occur relatively frequently in high resolution video. It is beneficial since large transforms are typically selected for coding such a smooth region.

2.5.4 Quantization

The basic principle for the quantization and de-quantization coefficients for large transforms is the same as that used in H.264/AVC, i.e. a scalar quantizer with dead-zone. Exactly the same QP range and corresponding quantization step have been used in the proposed codec.

2.6 In-loop filtering

In this proposal, in-loop filter is a combination of several spatial processes: deblocking filtering, extreme correction, adaptive loop filtering, band correction and content adaptive dynamic range adaptation. Each process is explained in more detail within the following sections.



Figure 2-21 in-loop filter combination

2.6.1 Deblocking filter

As in H.264/AVC, a deblocking filter is employed in this proposal. The boundary strength decision is the same as in H.264/AVC, if the CU boundary is regarded as equivalent to a macroblock boundary. That is, coding modes of adjacent blocks, quantization step size, and the steepness of the luminance gradient between blocks are considered in the same way.

However, as this proposal supports variable sizes of CU, PU and TU while the boundary decision was designed for fixed sizes of macroblock and partition in H.264/AVC, a new boundary decision for this flexible structure has been developed. It is known that prediction units cause prediction error and transform units cause quantization error in lossy coding. These errors result in blocking artifacts. To compensate for these artifacts, we need to consider prediction boundaries and transform boundaries together. Unlike H.264/AVC, it is not guaranteed that the transform unit is always included in the prediction unit, thus the boundary for the deblocking process is chosen from the boundary of the smaller unit. The remaining process is identical to the deblocking filter in H.264/AVC, i.e. it uses the same filters and follows the same boundary strength decision.

However, the overall complexity is reduced in many cases, since the flexible hierarchical structure of the proposal often causes large blocks to be used, which reduces the total number of boundary pixels to be filtered in comparison with H.264/AVC.

2.6.2 CU-synchronized adaptive loop filter (ALF)

After extreme values are corrected, adaptive loop filtering is invoked. The purpose of this process is to further reduce the distortion between the original picture and the reconstructed picture caused by complex lossy coding. Filters minimizing the distortion for both luma and chroma components are calculated using the Wiener filter approach. After filters are applied in a frame or coding units, filter coefficients are explicitly sent in the bitstream. This adaptive loop filter increases reconstructed picture quality as well as reference picture quality for the next picture coding. Several adaptive loop filter schemes have been proposed: frame-based, block-based and quadtree-based [12] adaptive loop filters with different filtering control basis. The filtering control basis is usually obtained independently from the prediction structure, which results in a large amount of information being sent to the decoder and its inability to access prediction information in its process.

In this proposal, CU-synchronized ALF is presented. By utilizing the coding unit structure, the amount of control information to be sent is reduced. Due to the flexible coding unit structure from large to small block, coding unit boundary provides great flexibility to be used as control basis in the adaptive loop filter process. Since the optimal coding unit boundary obtained from previous coding stages reflects objects' boundaries having similar properties, like prediction and quantization error, a simple control scheme which infers filter control structure from the coding unit structure and the optimal control depth reduces the complexity of the optimal control map while achieving similar performance.

Once the control units for filtering are determined, a control flag indicating whether filtering is applied is sent to maximize filter performance and reduce complexity of filter application. Depending on the largest block size, the maximum depth, sequence property and coding parameters, the number of coding units can be too many to be used as filter control units itself. Thus the optimal control depth is sent to indicate the maximum depth (i.e., the smallest size) of a coding unit boundary to be used as filter control map.

2.6.3 Extreme correction (EXC) and band correction (BDC)

Both the adaptive loop filter and the deblocking filter can be considered as adaptive low-pass filtering. The deblocking filter works only for the block edge pixels whereas the adaptive loop filter reduces overall distortion by computing optimal filter coefficients. Both techniques are effective to reduce the distortion as well as to improve the subjective quality; however, they do not consider the local property of the contents, e.g. object edge. For this purpose, two further techniques, EXC and BDC, are proposed to reduce the distortion based on the pixel classification.

EXC tries to classify all pixels into several categories and compute correction value for each category based on the relationship between current pixel value and four neighboring pixels, left, above, right and bottom. Let P be the current pixel value. Table 2-10 shows the conditions used in the classification stage in EXC.

Table 2-10 Conditions used in the classification stage in EXC

Category	Condition	Note
0	P < 4 neighbors	Local min
1	P < 3 neighbors && P = 4th neighbor	Object edge
2	P < 3 neighbors && P > 4th neighbor	Object edge
3	P > 3 neighbors && P < 4th neighbor	Object edge
4	P > 3 neighbors && P = 4th neighbor	Object edge
5	P > 4 neighbors	Local max

For pixels in each category, the mean difference between the reconstructed signal and the original signal is computed and transmitted to the decoder. EXC is useful to correct the unusual values (category 0 and 5) or the pixels near the object edge (category 1, 2, 3 and 4).

To reduce the number of conditional operations, the following pseudo-code can be used to find the category:

```
ExtremeType = 0;
if (Rec[x][y] > Rec[x-1][y]) ExtremeType++;
if (Rec[x][y] < Rec[x-1][y]) ExtremeType--;
if (Rec[x][y] > Rec[x+1][y]) ExtremeType++;
if (Rec[x][y] < Rec[x+1][y]) ExtremeType--;
if (Rec[x][y] > Rec[x][y-1]) ExtremeType++;
if (Rec[x][y] < Rec[x][y-1]) ExtremeType--;
if (Rec[x][y] > Rec[x][y+1]) ExtremeType++;
if (Rec[x][y] < Rec[x][y+1]) ExtremeType--;
```

The resultant ExtremeType has distinct values of -4, -3, -2, 2, 3 and 4 for each category, respectively, thus 8 comparisons for each pixel are needed for the EXC.

In BDC, pixel classification is simply done according to the value ranges. The main motivation of this is to equalize two different PDF's of the reconstructed signal and the original signal. The p most significant bits of the pixels are used for the classification. For example, if p = 4, a total 16 bands are defined. For each band, the mean difference is computed and transmitted to the decoder.

2.6.4 Content adaptive dynamic range (CADR)

The CADR provides high accuracy for internal processes by exploiting given luma or chroma sample ranges. Unlike internal bit depth increase (IBDI) technique, CADR itself doesn't require bit depth increase. This makes it preferable for hardware implementations. In addition, CADR can be used with IBDI to further improve the possible accuracy.

The CADR process can be defined by two parameters: MaxOrg and MinOrg which represent the minimum and maximum values of the input source, respectively. In principle, those parameters can be estimated on a slice-by-slice basis to maximize the effect, but this proposal simply uses the safe range between [16, 235] from the definition in BT.709 in order to reduce the encoder complexity.

Once the parameters are given, new values F(x) can be simply generated by

$$F(x) = MaxOrg * (x - MinOrg) / (MaxOrg - MinOrg)$$

It should be noted that the conversion can be simply implemented by table look-up. Figure 2-22 illustrates a mapping function F(x) when the range of [16, 235] is used.

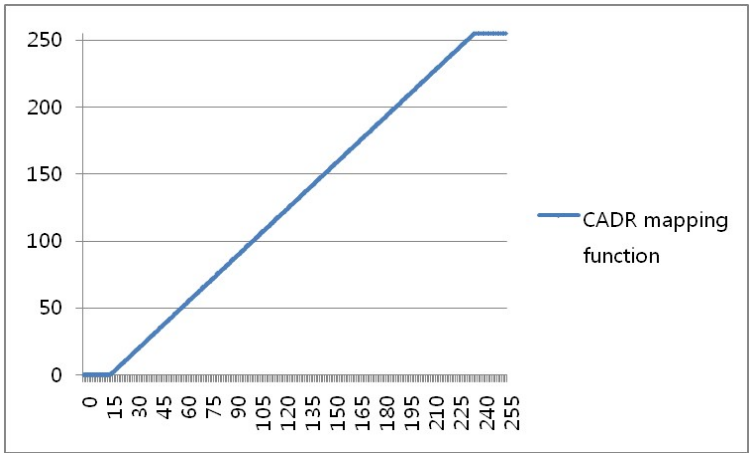


Figure 2-22 Example of a mapping function of CADR

Table 2-11 shows an example of the performance related to the various combinations of IBDI and CADR, using a one second segment of the Class D sequences. These results show that both tools are useful, although CADR has the advantage of requiring only 8-bit bit-depth.

Table 2-11 Performance of various combinations of IBDI and CADR (1s, GOP 8)

Sequence	CADR range	IBDI off, CADR on BD-rate (%)	IBDI (4-bit) on, CADR off BD-rate (%)	IBDI (4-bit) on CADR on BD-rate (%)
BasketballPass	[16, 235]	1.86	1.28	3.92
BQSquare	[16, 235]	4.34	1.86	5.90
BlowingBubbles	[16, 235]	2.09	1.27	3.74
RaceHorses	[16, 235]	1.72	1.20	3.49

2.7 Entropy coding

2.7.1 Syntax-based context-adaptive binary arithmetic coding (SBAC)

Entropy coding, which improves the coding performance by exploiting the statistical redundancy in the syntax elements of a video sequence, is an essential element of all current video codecs, including this proposal. Variable-length Coding (VLC) methods and their context-adaptive variants, e.g. the Context-Adaptive Variable-Length Coding (CAVLC) of H.264/AVC, have been widely used in the past due to their low complexity and ease of implementation. However, VLC-based methods are inherently sub-optimal in terms of coding performance, and it is hard to make them adapt to non-stationary symbols. Arithmetic coding is known to outperform VLC-based methods and can be well adapted to the varying statistical properties of the input symbols. However, its high complexity has hindered its adoption in video coding standards. Recently, due to the needs for higher coding efficiency and the advances in available computing power of codec platforms, arithmetic coding methods, e.g. Syntax-based Arithmetic Coding (SAC) of H.263 and Context-Adaptive Binary Arithmetic Coding (CABAC) of H.264/AVC, are adopted and widely used in the state-of-the-art video codecs.

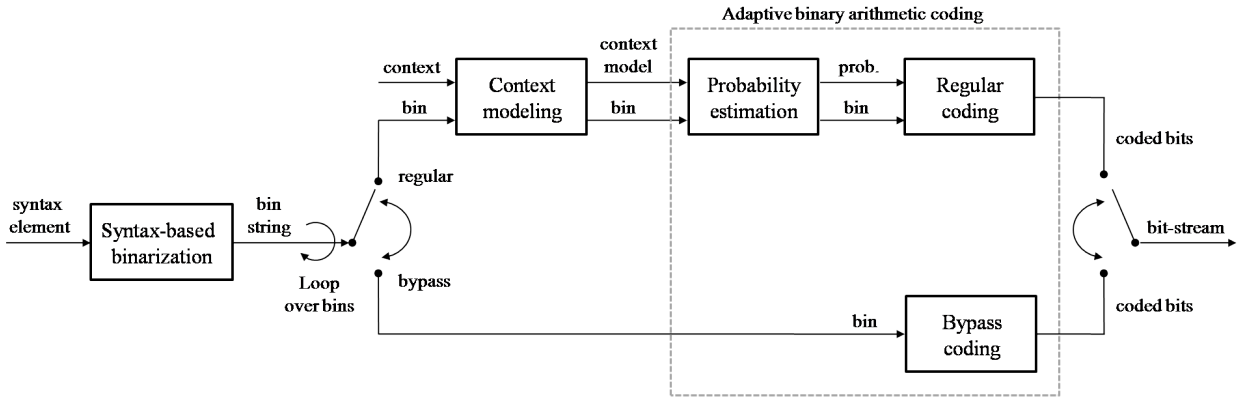


Figure 2-23 Block diagram of SBAC encoder

In this proposal, Syntax-based context-adaptive binary arithmetic coding (SBAC) is used for the entropy coding of syntax elements. SBAC is an adaptive binary arithmetic coding method with context modeling, which can provide high coding efficiency for various syntax elements with varying statistical properties. The coding core of SBAC is based on the arithmetic coding method specified in Annex D of JPEG still image coding standard, although it has been re-designed to meet the requirements of the proposal. Furthermore, a number of features desirable for video codec, e.g. bypass coding mode to reduce complexity, are also supported.

As shown in Figure 2-23, the encoding process of SBAC is composed of four major steps – syntax-based binarization, context modeling, probability estimation, and binary arithmetic coding. Even though the overall structure of SBAC is similar to that of the well-known CABAC, the details of each step are different from those of CABAC, and these differences lead to better coding performance.

1) ***Syntax-based binarization*** – to limit complexity and avoid context dilution problems, SBAC uses a binary arithmetic coding core. Since only binary values can be coded using SBAC, any non-binary syntax elements such as motion vector difference values must be binarized. For each syntax element, SBAC chooses an appropriate binarization method which can provide the best possible coding performance. For example, for binarization of the partition size, a modified unary code taking into accountthe geometry of the partition has been designed. Note that, when a given syntax element is binary, the binarization process is skipped.
2) ***Context modeling*** – the performance of an arithmetic coding method is dependent on the accuracy of the probability model that the coder uses. SBAC uses a context model, which is a probability model for one or more binary values to be coded, to provide an accurate estimate of the probability for each binary value. For each bin of each syntax element, a set of one or more context models is provided, and one of them is selected based on a given context and the bin value currently being coded. The rule for the selection of the context models is defined for each bin of each syntax element, and designed to provide the most relevant probability estimate to the coding engine.
3) ***Probability estimation*** – in SBAC, each context model consists of the most probable symbol (MPS) and the probability of the least probable symbol (P_{LPS}), and they are updated at each coding process based on the previous state of the context model and the coded binary value. For accurate and fast estimation, a finite state machine (FSM) based method is used for probability estimation and update.
4) ***Binary arithmetic coding*** – the binary arithmetic coding engine of SBAC is based on the arithmetic coding method specified in Annex D of the JPEG still image coding standard. Given a binary value to be coded and the corresponding context model, the binary arithmetic coding engine performs entropy coding based on its internal parameters. To improve the coding performance and meet the requirements of the video codec, the coding engine has been re-designed. For example, the table for probability estimation has been completely modified to be more accurate and adapt faster . The multiplication-free design has also been modified to improve performance while maintaining low complexity. A bypass coding mode is used for syntax elements with approximately equi-probable property, such as sign of motion vector difference. Note that, when the bypass coding mode is used, context modeling and probability estimation are skipped.

2.7.2 Adaptive coefficient scanning (ACS)

There are several elements in the codec which should be adapted to maximize the effect of large transforms. The transform coefficient scanning method is one of them. Since large transforms frequently have very sparse coefficient distributions, a fixed zigzag based scanning method often results in long coefficient scanning lists including many zeroes. The Adaptive Coefficient Scanning technique (ACS) was designed to improve the coding performance of coefficients of large transforms.

For each transform unit, the ACS index is parsed. Based on the ACS index, the scanning method is chosen from 0: conventional zigzag scan, 1: horizontal scan, 2: vertical scan. The ACS index is only necessary when there are non-DC coefficients.

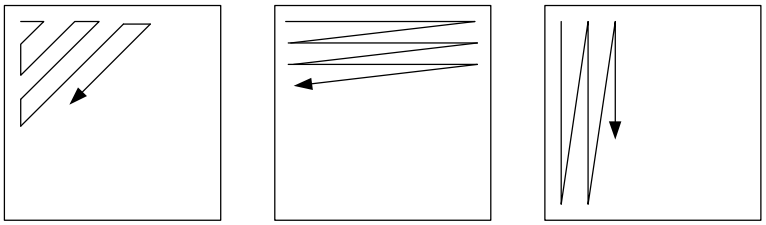


Figure 2-24 Zig-zag, horizontal and vertical scanning

3 Compression performance discussion

3.1 Objective versus subjective compression performance

The coding tools in this proposal represent an extension of the type of tools (prediction + DCT-based transform + entropy coding) used in H.264/AVC. As a result, there is not expected to be any really major difference with H.264/AVC in the way in which subjective and objective assessments compare.

One exception to this may be in the subjective visibility of blockiness, where the use of a large transform in the proposal can be expected to generate less blocky images, since the total number of edge pixels is decreased.

3.2 Constraint set 1 configuration relative to Alpha anchor

For satisfying constraint set 1, a typical hierarchical-B structure of GOP size 8 was used as specified in the joint CfP document [13], thus the number of frames required to decode a given frame is exactly the same as the Alpha anchor. Further information for test configuration is as follows with bold marks indicating differences compared to the Alpha anchor.

- Hierarchical B pictures IbBbBbBbP (8) coding structure – each picture uses 2 reference pictures for forward prediction and 2 reference pictures for backward prediction.
    - Open GOP structuring with an intra picture every 24, 32, 48 and 64 pictures for 24 fps, 30 fps, 50 and 60 fps sequences, respectively
    - QP scaling: QP (I picture), QP+1 (P picture), QP+2 (first B layer), QP+3 (second B layer), QP+4 (third B layer)
    - Same random access capability to Alpha anchor
- Flat quantization weighting matrices
- RD optimization enabled
- RDOQ enabled (fast mode, NUM=1)
- Adaptive rounding disabled
- **Weighted prediction disabled**
- Fast motion estimation (range 128x128)
- **Picture-level multi-loop encoding is disabled**
- **Largest block size of 128x128 is used except for Class D which uses 64x64**
- **12-tap DIF for luma and 6-tap DIF for chroma are used**

Table 3-1 shows the summarized performance of the proposed codec relative to the Alpha anchor. The average compression performance improvement over all sequences is 39.49%. Class B (1080p resolution) achieves 43.29% bit saving on average, with BQTerrace as the best sequence, showing 54.06% performance improvement. RaceHorses in Class D shows the lowest performance improvement, at 27.08%.

Table 3-1 Summarized performance relative to Alpha anchor

Class	Resolution	Seq. No	Seq. Name	BD-rate (%)
A	4K	S01	Traffic	40.59
		S02	PeopleOnStreet	29.84
B	1080p	S03	Kimono	44.42
		S04	ParkScene	32.96
		S05	Cactus	40.78
		S06	BasketballDrive	44.21
		S07	BQTerrace	54.06
C	WVGA	S08	BasketballDrill	43.09
		S09	BQMall	39.69
		S10	PartyScene	41.13
		S11	RaceHorses	35.86
D	WQVGA	S12	BasketballPass	30.55
		S13	BQSquare	54.19
		S14	BlowingBubbles	33.97
		S15	RaceHorses	27.08
Summary				
A				35.21
B				43.29
C				39.94
D				36.45
All				39.49

3.3 Constraint set 2 configuration relative to Beta and Gamma anchors

For satisfying constraint set 2, only the first frame is coded as intra frame and backward temporal prediction is prohibited in rest of frames, whilst maintaining the same display and coding order as the anchors. A hierarchical P structure with GOP of 4 was used, however IPPP structure was used to keep the bit-rate fluctuation as small as possible when the performance difference between two temporal structures is not significant. Detailed information for the test configuration is as follows with bold marks of differences compared the Beta anchor and Gamma anchor.

- No random access refresh requirement (a single I frame as the first picture)
- Hierarchical P pictures IpPp (GOP size of 4, with 3 temporal P picture layers, and no backward reference for inter prediction) coding structure – each picture uses at most 4 reference pictures for inter prediction
    - QP scaling: QP (I picture), QP+1 (first P layer), QP+4 (second P layer), QP+5 (third P layer)
    - Same structural delay constraints to Beta anchor
- **IPPP coding structure IPPP – no non-referenced frames are used**
    - **QP scaling: QP (I picture), QP+1 (first P layer), QP+2 (second P layer), QP+3 (third P layer)**
    - **Same structural delay constraints to Gamma anchor**
- Flat quantization weighting matrices
- RD Optimization enabled
- RDOQ enabled (fast mode, NUM=1)
- Adaptive rounding disabled
- **Weighted prediction disabled**
- Fast motion estimation. (range 128x128)
- **Picture-level multi-loop encoding is disabled**
- **Largest block size of 128x128 is used except Class D which uses 64x64**
- **12-tap DIF for luma and 6-tap DIF for chroma are used**

Table 3-2 shows the summarized performance of the proposed codec relative to the Beta anchor sequences. Average performance improvement over all sequences is 39.48% which is almost the same improvement as for Constraint Set 1. In particular, Class B (1080p resolution) achieves 43.66% bit saving on average. The best performance improvement over a Class is obtained with Class E, at 47.74%. BQTerrace shows the greatest performance improvement for a single sequence, at 60.62%. RaceHorses in Class D shows the lowest performance improvement, with 21.59% bit saving.

Table 3-2 Summarized performance relative to Beta anchors

Class	Resolution	Seq. No	Seq. Name	BD-rates (%)	Temporal structure
B	1080p	S03	Kimono	44.45	IPPP
		S04	ParkScene	31.18	Hier-P
		S05	Cactus	34.74	Hier-P
		S06	BasketballDrive	47.29	IPPP
		S07	BQTerrace	60.62	Hier-P
C	WVGA	S08	BasketballDrill	36.82	Hier-P
		S09	BQMall	37.06	IPPP
		S10	PartyScene	40.45	Hier-P
		S11	RaceHorses	27.99	IPPP
D	WQVGA	S12	BasketballPass	26.93	IPPP
		S13	BQSquare	51.87	Hier-P
		S14	BlowingBubbles	28.24	Hier-P
		S15	RaceHorses	21.59	IPPP
E	720p	S16	Vidyo1	48.38	Hier-P
		S17	Vidyo3	46.26	Hier-P
		S18	Vidyo4	47.79	Hier-P
Summary					
B				43.66	
C				35.58	
D				32.16	
E				47.48	
All				39.48	

Table 3-3 shows the summarized performance of the proposed codec relative to the Gamma anchor sequences. The average performance over all sequences is about 15% higher than for the Beta anchors.

Table 3-3 Summarized performance relative to Gamma anchors

Class	Resolution	Seq. No	Seq. Name	BD-rate (%)
B	1080p	S03	Kimono	58.01
		S04	ParkScene	47.92
		S05	Cactus	52.48
		S06	BasketballDrive	59.15
		S07	BQTerrace	72.87
C	WVGA	S08	BasketballDrill	54.37
		S09	BQMall	50.75
		S10	PartyScene	59.80
		S11	RaceHorses	36.97
D	WQVGA	S12	BasketballPass	37.70
		S13	BQSquare	74.25
		S14	BlowingBubbles	49.17
		S15	RaceHorses	28.62
E	720p	S16	Vidyo1	60.98
		S17	Vidyo3	59.55
		S18	Vidyo4	62.63
Summary				
B				58.09
C				50.47
D				47.44
E				61.05
All				54.08

4 Complexity analysis

4.1 Encoding time and measurement methodology

For encoding time measurement, a clustered system with multiple CPUs and RAMs was used. Each CPU has at most similar clock-speed with AMD Opteron 2.6GHz. The 32 bit executable is generated by GCC 4.12. The execution time is measured on RedHat Linux using intrinsic API functions of Linux.

Table 4-1 Results of encoding time measurement for constraint set 1

Class	Resolution	Seq. No	Seq. Name	Enc. Time (h/QP)
A	4K	S01	Traffic	60.42
		S02	PeopleOnStreet	67.55
B	1080p	S03	Kimono	76.53
		S04	ParkScene	58.28
		S05	Cactus	113.01
		S06	BasketballDrive	113.01
		S07	BQTerrace	96.00
C	WVGA	S08	BasketballDrill	24.02
		S09	BQMall	28.46
		S10	PartyScene	19.82
		S11	RaceHorses	24.68
D	WQVGA	S12	BasketballPass	5.94
		S13	BQSquare	4.63
		S14	BlowingBubbles	3.83
		S15	RaceHorses	4.47
Summary				
A				63.99
B				91.37
C				24.24
D				4.72

Table 4-2 Results of encoding time measurement for constraint set 2

Class	Resolution	Seq. No	Seq. Name	Enc. Time (h/QP)
B	1080p	S03	Kimono	51.41
		S04	ParkScene	43.49
		S05	Cactus	97.98
		S06	BasketballDrive	90.07
		S07	BQTerrace	104.21
C	WVGA	S08	BasketballDrill	19.69
		S09	BQMall	17.91
		S10	PartyScene	14.23
		S11	RaceHorses	18.06
D	WQVGA	S12	BasketballPass	3.91
		S13	BQSquare	6.19
		S14	BlowingBubbles	2.46
		S15	RaceHorses	2.87
E	720p	S16	Vidyo1	27.06
		S17	Vidyo3	25.36
		S18	Vidyo4	28.63
Summary				
B				77.43
C				17.47
D				3.86
E				27.02

4.2 Decoding time and measurement methodology

For the decoding time measurements, the test platform is a PC based on Intel Core2 Quad CPU Q9400 (2.66GHz) and 2GB RAM. The 32-bit executables of JM17.0 and the proposed codec were generated by Intel compiler 11.0.072. The execution time was measured on Windows XP Professional SP3 using intrinsic API functions of Windows. As recommended in the CfP document and the reflector, YUV writing was enabled. Only one core was utilized for measuring the decoding time while other cores remain empty. Table 4-3 shows the decoding time for decoding the submitted bitstreams under constraint set 1 and constraint set 2.

As shown in the table, the average decoding time of proposed codec takes 13% less than JM17.0 for constraint set 1 and 13% less for constraint set 2. It is clear that this does not mean that the proposed codec actually has 13% less computational complexity in all implementations, however it is safe to say that the proposed codec at least does not require unacceptable computational power and the software level seems to be mature.

Table 4-3 Results of decoding time measurement for constraint set 1

Class	Resolution	Seq. No	Seq. Name	JM17.0 Time (sec/QP)	JM17.0 Fps	Prop. Time (sec/QP)	Prop. Fps	Rel. Time (Prop/JM)
A	4K	S01	Traffic	137.04	1.09	83.39	1.80	0.61
		S02	PeopleOnStreet	136.91	1.10	91.85	1.64	0.67
B	1080p	S03	Kimono	114.98	2.09	74.25	3.25	0.65
		S04	ParkScene	112.53	2.13	70.84	3.43	0.63
		S05	Cactus	224.86	2.23	118.67	4.25	0.53
		S06	BasketballDrive	238.26	2.52	162.21	3.71	0.68
		S07	BQTerrace	288.85	2.08	161.88	3.73	0.56
C	WVGA	S08	BasketballDrill	29.21	17.12	21.33	23.66	0.73
		S09	BQMall	35.42	16.95	25.42	23.95	0.72
		S10	PartyScene	30.00	16.68	22.76	22.50	0.76
		S11	RaceHorses	18.31	16.41	17.47	17.63	0.95
D	WQVGA	S12	BasketballPass	4.67	107.80	6.57	78.12	1.41
		S13	BQSquare	5.55	109.05	8.09	75.20	1.46
		S14	BlowingBubbles	4.76	106.71	6.12	83.46	1.28
		S15	RaceHorses	3.17	95.63	5.19	59.78	1.64
Summary								
A				136.98	1.10	87.62	1.72	0.64
B				195.89	2.21	117.57	3.67	0.61
C				28.23	16.79	21.75	21.93	0.79
D				4.54	104.80	6.49	74.14	1.45
ALL								0.87

Table 4-4 Results of decoding time measurement for constraint set 2

Class	Resolution	Seq. No	Seq. Name	JM17.0 Time (sec/QP)	JM17.0 Fps	Prop. Time (sec/QP)	Prop. Fps	Rel. Time (Prop/JM)
B	1080p	S03	Kimono	108.53	2.21	75.92	3.18	0.70
		S04	ParkScene	107.89	2.23	64.35	3.79	0.60
		S05	Cactus	227.47	2.20	119.09	4.29	0.52
		S06	BasketballDrive	231.37	2.59	165.10	3.66	0.71
		S07	BQTerrace	276.88	2.17	161.49	3.73	0.58
C	WVGA	S08	BasketballDrill	27.90	17.95	22.84	22.22	0.82
		S09	BQMall	34.12	17.60	26.08	23.41	0.76
		S10	PartyScene	28.62	17.51	20.92	24.16	0.73
		S11	RaceHorses	17.77	16.93	19.32	16.07	1.09
D	WQVGA	S12	BasketballPass	4.47	113.16	6.85	75.20	1.53
		S13	BQSquare	5.30	114.04	7.62	79.50	1.44
		S14	BlowingBubbles	4.65	108.85	5.87	85.85	1.26
		S15	RaceHorses	3.17	96.43	5.78	54.11	1.82
E	720p	S16	Vidyo1	96.25	6.24	46.36	12.99	0.48
		S17	Vidyo3	94.33	6.36	45.16	13.35	0.48
		S18	Vidyo4	94.97	6.32	44.80	13.42	0.47
Summary								
B				190.43	2.28	117.19	3.73	0.62
C				27.10	17.50	22.29	21.47	0.85
D				4.40	108.12	6.53	73.67	1.51
E				95.18	6.31	45.44	13.25	0.48
ALL								0.87

4.3 Expected memory usage of encoder

The required memory of the proposed encoder is comparable to H.264/AVC. Although the size of several temporal buffers is increased according to the LCU size and the maximum depth, it is not a major impact compared to the reference memory size.

4.4 Expected memory usage of decoder

The required memory of the proposed decoder is comparable to H.264/AVC except the adaptive loop filter. Since the adaptive loop filter is applied to the reconstruction signal on the frame-basis, one additional frame buffer storing the reconstruction signal before applying the adaptive loop filter is required. However, because the number of filter taps is limited, the required buffer size can be reduced significantly with some increases of the buffer management complexity.

4.5 Complexity characteristics of encoder motion estimation and motion segmentation selection

4.5.1 Motion estimation

In this proposal, motion vector accuracy normally ranges from integer to 1/4 pel with the addition of eight 1/12 pel accuracy positions. Therefore, in principle it covers an area 9 times as large as H.264/AVC motion grid.

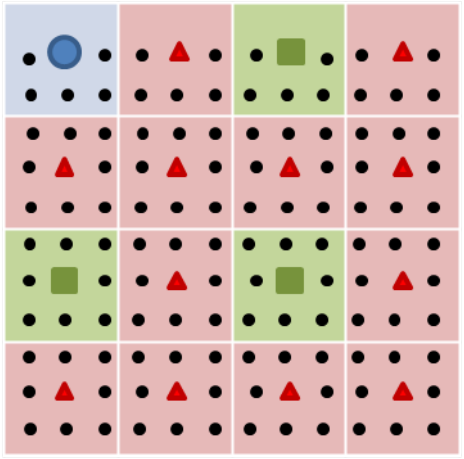


Figure 4-1 Fractional pel grid

However, even though its coverage is much larger than H.264/AVC, the complexity of motion estimation is only marginally increased because motion estimation for 1/12 pel accuracy can be realized by 9 refinements around the best 1/4 accuracy position. Luma SAD was used for the integer motion search and luma hadamard for the fractional motion search.

4.5.2 Motion segmentation selection

Motion segmentation (or motion partition) is done by the typical rate-distortion optimization process. SSE was used to compute the distortion for motion segmentation. Compared to H.264/AVC, four shapes of asymmetric motion partition are added and the supported hierarchy level is increased. As a result, encoder complexity in this implementation is higher than H.264/AVC. However, large block search costs can be ameliorated in real implementations by well-known techniques (phase correlation, hierarchical block matching, etc.).

4.6 Complexity characteristics of decoder motion compensation

4.6.1 Interpolation process

As already stated in Section 2.3.4.3, the proposed DIF is a non-cascaded interpolation scheme, i.e. only one spatial filtering is performed for all possible positions, which can be implemented by a multiplication-free design. Thus, the 6-tap DIF has the smaller number of operations compared to H.264/AVC since pixels at the quarter-pel position can be generated by one 6-tap filtering operation whereas H.264/AVC uses the combination of 6-tap and bi-linear filters.

The 12-tap DIF filter requires two times as many operations as the 6-tap DIF, however it provides similar performance to the well-known adaptive interpolation filter (AIF) scheme whilst it has much less complexity and is more suitable for hardware implementation.

High accuracy motion (HAM) also uses one spatial filtering to get 1/12 accuracy position, so the complexity characteristic is similar to the DIF. It's interesting to see that the complexity of DIF is virtually the same at any arbitrary accuracy point. This property allows the design of motion compensation processes based on arbitrary motion accuracy without increasing the complexity according to the supported motion accuracy.

4.6.2 Number of motion compensations

Many operations such as motion vector access, motion vector predictor computation and reference position computation are performed once for each motion compensation unit. Therefore, the average number of motion compensations is very important in reducing this kind of complexity. Table 4-5 shows the average number of motion compensations collected from the Class B streams of constraint set 1. It is very clear that the proposed method requires only 1/3 of the number of motion compensations compared to the Alpha anchor, mainly due to the use of large motion compensation units, which naturally results in the complexity reduction. It also means that the memory bandwidth required to access the motion information can be reduced significantly.

Table 4-5 Number of motion compensations in Class B streams

Sequence	Alpha anchor	Proposed method	Ratio
Kimono	1896948	577465	30.4%
Parkscene	2056280	811951	39.4%
Cactus	4134668	1420699	34.3%
BasketballDrive	3865489	1159876	30.0%
BQTerrace	5198921	1488056	28.6%
Average	-	-	32.5%

4.7 Complexity characteristics of encoder intra-frame prediction type selection

For ADI, the number of directions depends upon the PU splitting. For the sizes of 4x4 and 8x8, 9 directions are used whereas 33 directions are supported for the sizes of 16x16 and 32x32. For the sizes of 64x64 and 128x128, only 5 directions are used. Once the PU size is given, RD comparison is performed for all possible directions.

In addition to the ADI predictions, the encoder performs pixel-based template matching and compares the result with the best ADI mode in terms of coding efficiency.

After ADI and pixel based template matching have determined the best luma prediction mode, the MPI decision is made. This determines whether the best performance is gained by applying the 4 pixel smoothing filter or not.

For chroma samples, the number of prediction modes is determined by the best prediction modes for luma samples. If PTM mode is chosen, or the best ADI modes are larger than DC (2), 5 prediction modes are supported. Otherwise, only 4 modes are used for chroma samples. For chroma predictions, the CCCP process replaces the DC prediction. Although CCCP performs region segmentation for luma samples, the process is not computationally demanding compared with other segmentation based algorithms, since it uses a simple thresholding approach.

4.8 Complexity characteristics of decoder intra-frame prediction operation

Where an ADI mode is signaled, the decoder performs the appropriate directional prediction by extrapolating reconstructed pixels based on the appropriate (dx, dy) direction. As a result, prediction of significant numbers of pixels in large blocks can be achieved by copying from referenced pixels.

While ADI is generally applied for most prediction units, pixel based template matching and CCCP are selected for only a small portion of prediction units. Therefore, although more complex, their effects on total decoder complexity are not as large as ADI.

The complexity of the pixel based template matching can be described in a similar way to motion estimation. Full integer pel search is applied in the block. For each pixel the search range is up to 27 pixels and the size of template is 3 pixels (refer to Figure 2-18).

For luma samples with MPI mode =1, the 4 pixel based smoothing operation is applied for each pixel after the first intra prediction stage.

For chroma samples which have DC prediction mode, CCCP is applied. The CCCP consists of 1) bilinear down-sampling of reconstructed luma 2) calculation of mean values of down-sampled luma samples, 2) thresholding by mean value 3) assigning the context pixels based on the thresholding results, 4) smoothing by 3x3 averaging filter. Averaging by 3x3 filter can be efficiently implemented by 3-tap averaging in horizontal direction following by 3-tap averaging in vertical direction. Also, the result of summation can be shared for neighbor pixels, resulting in further complexity reduction.

4.9 Complexity characteristics of encoder transforms and transform type selection

For a given PU splitting, there could be two kinds of TU sizes. Between the two candidates, the best TU size is determined to be the one which produces smaller rate-distortion costs. The complexity of the transform itself is described in Section 4.10 in detail.

For intra CU, ROT is followed by the DCT. In the process to determine the best TU size, the optimal ROT matrix between the 4 possible matrices is determined to be the one which provides the smallest RD cost. To minimize the related complexity for finding the best ROT matrix, the ROT search is invoked only if the best prediction mode is intra-type. The complexity of ROT itself is described in Section 4.10.2.

4.10 Complexity characteristics of decoder inverse transform operations

4.10.1 Large size DCT

In this section, an analysis of the complexity of inverse transforms based on the number of operations is presented. First, the number of operations required for a 16x16 block is summarized in Table 4-6. In this table, + and >> mean addition and shift operation. For a 16x16 block, sixteen 4x4 transforms or four 8x8 transforms are required to process entire block.

Table 4-6 Number of operations per 16x16 block

Transform size	Operation	Number / 16x16	Total operations
4x4	+ >>	256 64	320
8x8	+ >>	256 80	336
16x16	+ >> *	148 52 88	288 (with 88 mult.)

To make the transform multiplication-free, simple conversions rule can be applied. Since the constant is dyadic rational, it can be factorized into the addition and/or subtraction of several dyadic rationals which have 1 as a numerator. For example, 3/8 can be factorized into 1/4 + 1/8 = (1 + (1<<1))>>3. So in this case 1 multiplication and 1 shift can be calculated by only 2 shifts and one addition without loss. Table 4-7 shows the multiplication-free results.

Table 4-7 Number of operations per 16x16 block, multiplication-free 16x16 transform

Transform size	Operation	Number / 16x16	Total operations
4x4	+ >>	256 64	320
8x8	+ >>	256 80	336
16x16	+ >>	300 236	536

The 32x32 transform and 64x64 transform are constructed in the similar manner to the 16x16 transform described above. The number of operations per 64x64 block is summarized in Table 4-8.

Table 4-8 Number of operations per 64x64 block

Transform size	Operation	Number / 64x64	Total operations	Ratio vs. 8x8
4x4	+ >>	4096 1024	5120	0.95
8x8	+ >>	4096 1280	5376	1.00
16x16	+ >> *	2368 832 1408	4608 (with 1408 mult.)	-
16x16 multi. free (multi. free)	+ >>	4800 3776	8576	1.60
32x32	+ >> *	1552 528 928	3008 (with 928 mult.)	-
32x32 (multi. free)	+ >>	3872 3096	6968	1.30
64x64	+ >> *	964 324 584	1872 (with 584 mult.)	-
64x64 (multi. free)	+ >>	2832 2368	5200	0.97

Note that the total number of operations per 64x64 block may become smaller when the transform size becomes larger, as shown in Table 4-8. In particular, the 64x64 multiplication-free transform requires only a similar number of operations as a 4x4 integer transform. Although 16x16 and 32x32 transforms require about 60% and 30% more operations respectively compared to the conventional 8x8 integer transform, the overall impact on the decoder complexity is not large since the portion of the inverse transform complexity among total decoder complexity is generally limited.

4.10.2 ROT

The ROT operation is two matrix multiplication operations. To minimize the complexity burden, ROT is applied only for the low-frequency elements which size is limited to 8x8 for any transform size. The total number of operations required for the inverse ROT of size 4x4 and 8x8 is represented in Table 4-9.

Table 4-9 Number of operations for inverse ROT

ROT size	Operation	Number	Total operations
4x4	+ >> *	12 0 18	30 (with 18 mult.)
8x8	+ >> *	48 0 72	120 (with 72 mult.)

4.11 Complexity characteristics of encoder quantization and quantization type selection

Only one type of quantization process is used in the proposed codec, i.e. no quantization matrix is used. In addition, the RDOQ process only uses one fixed QP value according to the anchor configuration.

4.12 Complexity characteristics of decoder inverse quantization

There is no specific difference compared to the H.264/AVC since they share the same scalar dequantizer scheme.

4.13 Complexity characteristics of encoder in-loop filtering type selection

4.13.1 Adaptive loop filter

For the luma component, there are two kinds of filter application control: a frame-based one and a unit-based one. Given the fixed tap which is 5x5 in the proposed codec's encoder, the optimal control structure is first estimated. For all possible control depths where a control depth is from 0 to the max CU depth – 1, the unique control map is generated. For example, if the max CU depth is set to 5, then there are only five control maps to be tested. For every unit in a control map, the control flag is determined by comparison of the original distortion and the new distortion where the new distortion is set to the difference between the original signal and the filtered reconstruction signal. The Wiener filter is re-estimated several times with units which prefer to be filtered. In this proposal, the Wiener filter is re-estimated up to 3 times by the encoder configuration. Because filtering is determined for each CU or merged CU, the correlation can be reused if it is calculated and stored for each smallest CU to reduce the complexity of the Wiener filter estimation. Once the best control map, control flags and the filter are selected for unit-based control, its RD cost is compared to the RD cost of frame-based control.

The optimal filter length is estimated with the selected control units and flags in the previous stage. In this proposal, there are three filter taps: 5x5, 7x7 and 9x9. For low complexity, this process is omitted in the proposed encoder when a slice type is non-referenced B.

For chroma components, filtering is only frame-based. Furthermore, its process is not invoked for non-referenced B, in order to reduce encoder complexity. Each chroma component can be applied or not independently and only one filter is shared for both chroma components. If one chroma component is decided not to be applied with the initially estimated filter, then the Wiener filter for the other chroma component is re-designed. For simplicity, only one filter tap is supported for chroma components and that is 5x5.

The optimal Wiener filter is usually estimated in floating type. It loses the accuracy when it is approximated and quantized to the limited integer type. But to reduce the overhead of filter coefficients, quantization is necessary. As many Wiener filters are estimated for a slice with multiple loop tests or different conditions, as there are many chances to find the quantized filter optimally to compensate for quantization loss. But consequently, it increases encoder complexity as a result of the many tests. As the control structure has the freedom to be generated in an arbitrary shape, it is important to limit the number of test cases in the encoder to limit the complexity of the task. In this proposal, it is proposed to use the CU boundary to generate a control map. The CU boundary is already a clustered prediction error region and quantization error region, and furthermore, due to its hierarchical representation, the proposed adaptive loop filter can control the tradeoff between filter efficiency and filter overhead. So with relatively few trials, the encoder can produce an efficient filter and the control map.

4.14 Complexity characteristics of decoder in-loop filtering operations

4.14.1 Deblocking filter

Although we use the same process defined in the H.264/AVC, the use of large block sizes can effectively reduce the complexity of the deblocking filter. Table 4-10 shows the average number of deblocked pixels in Class B sequences for constraint set 1. As shown in the table, the number of deblocked pixels can be reduced by 41% in average, which means that the complexity of the deblocking filter can be reduced significantly in the proposed scheme.

Table 4-10 Average number of deblocked pixels in Class B of constraint set 1

Sequence	Number of deblocked pixels		Relative ratio
	Alpha anchor	Proposed	
Kimono1	145951397	70604005	0.48
ParkScene	86871747	54583037	0.63
Cactus	205011157	126507089	0.62
BasketballDrive	316322552	157577813	0.50
BQTerrace	132100148	96465482	0.73
Average	177251400	101147485	0.59

4.14.2 Adaptive loop filter

5x5 tap, 7x7 tap, and 9x9 tap Wiener filters with offset are used. Complexity for coding, parsing and applying the filters is reduced by exploiting the filter symmetries as follows [12] (each number corresponds to a tap value):

Table 4-11 Position of filter coefficients

(a) 9x9 tap

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
27	28	29	30	31	32	33	34	35
36	37	38	39	40	39	38	37	36
35	34	33	32	31	30	29	28	27
26	25	24	23	22	21	20	19	18
17	16	15	14	13	12	11	10	9
8	7	6	5	4	3	2	1	0

(b) 7x7 tap

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	23	22	21
20	19	18	17	16	15	14
13	12	11	10	9	8	7
6	5	4	3	2	1	0

(c) 5x5 tap

0	1	2	3	4
5	6	7	8	9
10	11	12	11	10
9	8	7	6	5
4	3	2	1	0

Since FIR filter is used with no data dependency, it makes parallelization easier.

Once it is determined to use the adaptive loop filter, it is applied to the whole frame or controlled by a unit. When the adaptive loop filtering is controlled by a unit, it is applied, or not, according to a binary partition map. This map is derived from the coding unit boundaries to reduce metadata and complexity. The partition map can be rendered coarse or fine by a filter control depth parameter which is signaled, and which lies between the lowest and the highest coding unit depth. Only coding unit boundaries whose depth is less than or equal to the filter control depth are used in deriving the partition map.

4.14.3 Extreme correction and band correction

Because the positions of reference pixels used for extreme correction are only those above, below, to the left and to the right, and the filter is FIR with no data dependency, this process is highly parallelizable. Extreme band correction needs no reference pixel, and so it can be merged into the adaptive loop filter. Both the filters used in extreme correction and extreme band correction work by replacing the target value by the respective value determined by some rule, which can be realized by a simple table lookup to reduce complexity.

4.15 Complexity characteristics of encoder entropy coding type selection

This proposal uses one entropy coding method, SBAC, thus there is no need to select entropy coding type.

4.16 Complexity characteristics of decoder entropy decoding operation

4.16.1 Syntax-based context-adaptive binary arithmetic coding

Overall, the complexity of SBAC is comparable to CABAC, since SBAC is also a context-adaptive binary arithmetic coding method. SBAC uses binarization techniques and bypass techniques similar to CABAC in complexity. The core binary coding engine uses a small FSM for probability updates, as does H.264/AVC. SBAC also uses a small number of shifts and additions for interval rescaling, unlike the LUT method in H.264/AVC JM implementation, but this is unlikely to affect relative performance.

4.16.2 Adaptive coefficient scanning

When the horizontal or vertical scanning methods are chosen as the best scanning method, the mapping process from the coefficient array index to the zigzag scanning index is not necessary. Thus, the complexity for the coefficient indexing can be reduced and also the memory bandwidth to access the zigzag mapping table can be saved.

4.17 Degree of capability for encoder parallel processing

Most of the parallel processing approaches for H.264/AVC encoders are based on the separate estimation of different level of units: GOP, slice or group of macroblocks. There is no particular problem in using typical H.264/AVC schemes for the proposed codec.

4.18 Degree of capability for decoder parallel processing

The proposed scheme provides the same level of parallelism compared to H.264/AVC. All inter-predicted units can be processed in parallel and each slice can be decoded independently. However, there is a significant benefit with the use of large size blocks for the processors supporting vector processing, e.g. single instruction multiple data (SIMD). Widely used SIMD techniques support the use of 128 bit register and even larger size register is now considered such as Intel advanced vector extension instructions (AVX) of 256 bit register.

Table 4-12 Average width of processing unit for various classes in constraint set 1

Class	Alpha anchor		Proposed		Ratio	
	MC	Intra	MC	Intra	MC	Intra
A	15.6	9.1	53.4	15.3	3.4	1.7
B	15.7	10.0	62.5	22.4	4.0	2.4
C	15.5	9.0	55.2	14.5	3.6	1.6
D	14.4	7.1	30.4	9.5	2.1	1.3
ALL					3.3	1.7

To maximize the usefulness of SIMD operations, the number of simultaneously processed pixels should be large. From the simple analysis as shown in Table 4-12, the average width of the motion compensation unit is 3.3 times longer than the Alpha anchor case and 1.7 times for the intra prediction unit. It means that the usefulness of larger size register can be improved significantly in the motion compensation and intra prediction processes in the proposed scheme.

4.19 Requirement for encoder frame-level multi-pass processing

There is no particular need to perform encoder frame-level multi-pass processing for the proposed scheme.

5 Algorithm characteristics

5.1 Random access characteristics

As described in Section 3.2, the constraint set 1 anchor GOP conditions condition were also satisfied by this proposal. Therefore, the number of frames required to decode any given frame is exactly same as the Alpha anchor, which means that the same random access interval of 1.1 sec can be achieved.

5.2 Delay characteristics

As described in Section 3.3, we have used both the typical hierarchical-P structure of GOP 4 and the IPPP structure to generate the bitstreams for constraint set 2. On the decoder-side, no particular reordering of pictures is necessary. Thus the structural delay characteristics of the proposal are exactly the same as that of the given anchors which satisfies constraint set 2 conditions.

To estimate the delay caused by bit-rate fluctuation, RVM was proposed in the Kyoto meeting. Table 5-1 summarizes the RVM values of the anchor and the proposed codec for each sequence class.

Table 5-1 RVM summary of the anchors and the proposed codec

Class	Resolution	Beta	Gamma	Proposed		Proposed (lambda fixed)	
		RVM	RVM	RVM	BD-rate (%)	RVM	BD-rate (%)
B	1080p	1.22	0.36	1.74	43.66		
C	WVGA	1.01	0.34	1.70	35.58		
D	WQVGA	1.12	0.38	1.63	32.16	1.12	31.19
E	720p	1.38	0.44	2.10	47.48		
ALL	-	1.17	0.38	1.77	39.48		

As shown in the table, RVM values of the proposal are increased compared to both anchors. We found that this effect comes from the fact that the compression efficiency is increased significantly higher in B-slices by the

proposed methods than in I- or P-slices. This increases the variance of the bit-rates, resulting in an increased RVM measurement. However, this situation can be avoided by a simple change of the lambda values of B-slices in the encoder. Fortunately, this kind of change does not result in any significant performance drop. In the test of Class D sequences, less than a 1% performance drop is observed when the RVM of the proposed codec is fixed to have the same value of Beta anchor as shown in Table 5-1.

6 Software implementation description

The proposed algorithms were implemented using standard C++ programming language.

7 Closing remarks

In the previous sections, all aspects of the proposed codec, including detailed algorithm descriptions, performance results and complexity analysis, have been discussed. On average, the proposed codec showed about 40% performance improvement over the Alpha and Beta anchor sequences. The complexity of the proposed encoder and decoder is similar to that of current standards.

In conclusion, we think that enough information has been provided to demonstrate the validity of this proposal and the reasons for it to be considered as a strong candidate to be used for the Test Model that will be used as the basis of the Core Experiments in the next phase of HVC standardization.

8 Patent rights declaration(s)

British Broadcasting Corporation (BBC) may have IPR relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).

Samsung may have IPR relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).

References

[1] K. H. Lee, E. Alshina, J. H. Park, W. J. Han and J. H. Min, "Technical considerations for Ad Hoc Group on New Challenges in Video Coding Standardization," ISO/IEC MPEG 85th meeting, M15580, Hannover, Germany, 2008.

[2] E. Alshina , K. H. Lee, W. J. Han and J. H. Park, Technical considerations for Ad Hoc Group on New Challenges in Video Coding Standardization," ISO/IEC MPEG 86th meeting, M15899, Busan, Korea, 2008.

[3] S. Naito, A. Matsumura and A. Koike, “Efficient coding scheme for super high definition video based on extending H.264 high profile”, VCIP 2006, Jan. 2006.

[4] S. Ma and C.-C. Jay Kuo, “High-definition video coding with super-macroblocks”, VCIP 2007, Jan 2007.

[5] P. Chen, Y.Ye and M. Karczewicz, “Video Coding Using Extended Block Sizes”, VCEG-AJ23, Oct 2008.

[6] Tomonobu Yoshino, Sei Naito and Shigeyuki Sakazawa, “Preliminary response for Draft Call for Evidence on High Performance Video Coding”, M16082, Feb 2009

[7] Shun-ichi Sekiguchi and Shuichi Yamagishi, “On coding efficiency with extended block size for UHDTV”, VCEG-C91, Jan 2009

[8] Joel Jung and Guillaume Laroche, “Competition-Based Scheme for Motion Vector Selection and Coding”, VCEG-AC06, July 2006

[9] Thomas Wedi, “Hybrid Video Coding Based on High-Resolution Displacement Vectors”, VCIP, Jan. 2001

[10] W-H. Chen, C. Harrison Smith and S. C. Fralick. “A Fast computational Algorithm for the Discrete Cosine Transform “, IEEE Trans. Communications, No. 9, Sept. 1977
[11] S. D. Kim and J. B. Ra, “An efficient motion vector coding scheme based on minimum bitrate prediction,” IEEE Trans. on Image. Proc, vol. 8, no. 8, pp. 1117–1120, Aug. 1999
[12] Takeshi Chujoh, Naofumi Wada and Goki Yasuda, “Quadtree-based adaptive loop filter”, VCEG-C181, Jan. 2009
[13] “Joint call for proposals on video compression technology,” ISO/IEC MPEG 91st meeting, N11113, Kyoto, Japan, Apr. 2010.