A Study on Hardware Architecture for H.265/HEVC Fast Mode Decision and Transform

January 2015

Wenjun ZHAO

Publication list

Transactions

1. W. Zhao, T. Onoye, and T. Song, “Hierarchical structure based fast mode decision for H.265/HEVC,” *IEEE Transactions on Circuits and Systems for Video Technology*, (in press).

Conference papers with referee

1. W. Zhao, T. Onoye, and T. Song, “Hardware architecture of the fast mode decision algorithm for H.265/HEVC,” In *Proc. of IEEE International Conference on Image Processing (ICIP)*, Oct. 2014.
2. W. Zhao, T. Onoye, and T. Song, “Hardware-oriented fast mode decision algorithm for intra prediction in HEVC,” In *Proc. of Picture Coding Symposium (PCS)*, pp. 109 - 112, Dec. 2013.
3. W. Zhao, T. Onoye, and T. Song, “High-performance multiplierless transform architecture for HEVC,” In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1668 - 1671, May 2013.

Conference papers without referee

1. W. Zhao, and T. Onoye, “A high-performance multiplierless hardware architecture of the transform applied to H.265/HEVC emerging video coding standard,” *IEICE Technical Report*, vol. 112, no. 207, SIS2012-18, pp. 11 - 16, Sep. 2012.

Abstract

This thesis mainly discusses the fast mode decision algorithms for the H.265/HEVC (high efficiency video coding). HEVC has incorporated a series of the state-of-the-art technologies and algorithms. These features help HEVC to achieve significantly high compression efficiency. However, these features also increase the computational complexity. In order to find the best encoding parameters (e.g. coding mode) for a certain block (comprised of luma and chroma components), a huge number of combinations of block sizes and candidate modes have to be checked, which is very time-consuming. Hence, in this work, a course of fast mode decision algorithms are proposed to accelerate the mode decision process. Moreover, the corresponding hardware architecture of the proposed fast decision algorithms as well as the hardware design of the transform of H.265/HEVC are proposed.

First, this thesis gives a brief introduction about the H.265/HEVC. The HEVC codec employs the well-known hybrid block-based coding framework, including advanced intra prediction with 35 modes, improved motion-compensation prediction with merge technique, newly added large-sized transform engine, and high-efficiency entropy coding tool. Moreover, the reconstructed pixels are filtered by the similar de-blocking and newly adopt sample adaptive offset filters before sent to the decoded picture buffer. Unlike the previous video coding standards, the HEVC adopts a flexible quadtree structure based block partition scheme that enables effective use of different block sizes during the prediction and transform coding processes. Two strategies aiming at overcoming the limitations of the parallelization approaches employed in H.264/AVC have been included in the HEVC, namely Tiles and wavefront parallel processing.

Second, this dissertation presents a course of low complexity fast mode decision algorithms. In order to skip some unlikely depths, the maximum depth information of a co-located block is referred to predict the depth of current block. To make a prediction, after encoding one frame, the depth information is saved, in order to enable later coded frames to refer to these data. In order to reduce the complexity introduced by saving the depth information, it is proposed in this dissertation that co-located LCU from the previous frame in encoding order will be used. Next, for a certain sized block, the motion character of inter prediction residual is analyzed to determine whether to terminate the current check or to skip over unnecessary modes and split the block into smaller sizes. In order to detect the motion character of each portion inside a block, we propose to divide the residual block and calculate the average and the sum of absolute difference over average. Two conditions are defined to terminate the mode check process or to skip current depth mode check process and move on to the next depth. Moreover, in order to skip some unlikely partition modes, a skip strategy is also proposed. To make compensation to a wrong split condition, a novel remedy process is introduced. After inter prediction, a hardware-oriented low complexity fast intra prediction algorithm is presented. The proposed algorithm adopts a fast discrete cross differences (DCD) to detect the dominate direction of the coding unit. Based on DCD information, only a subset of the 35 candidate modes are selected for the rough mode decision process. Moreover, four simple but efficient early termination strategies are proposed to terminate the RDO process properly.

Third, in this thesis, the corresponding hardware architectures of the proposed fast mode decision algorithms are proposed. In order to achieve a better compatibility, the proposed fast mode decision architectures are designed as an individual module that can be easily embedded into a common video codec for H.265/HEVC. A state machine based mode dispatch module for the depth prediction combined with the residual check algorithm is described. In this mode dispatch module, mainly 4 kinds of elements are contained according to its functional definition: information recording element, controlling and decision making element, interface element, and the core mode dispatcher. For the proposed state machine, there are 13 states are defined. Then, the hardware implementation of the proposed fast DCD algorithm and two previous works are discussed. Moreover, the complexity and performance of the proposed DCD algorithm is compared with previous works.

Finally, this dissertation describes a hardware architecture of the transform applied in HEVC. The proposed architecture can support a variety of transform sizes from 4x4 to 32x32. The hardware design proposed in this work focuses on low cost and high throughput. To achieve such objectives, some simplification strategies are adopted during the implementation, such as reusing part of the structure of the larger sized transform for smaller sized transform, and turning multiplication by constant into shift and sum operations. The transform architecture proposed in this dissertation is implemented in the form of pipeline structure. Moreover, a high-performance transposition memory is proposed to store and transpose the intermediate data between the 1-D and 2-D transform.

As a conclusion, in this thesis, a course of fast mode decision algorithms and its corresponding hardware architectures as well as the hardware design of the transform of H.265/HEVC are proposed.

Acknowledgments

First of all, I would like to express my deepest gratitude to Professor Takao Onoye in Osaka University for providing me a precious opportunity and an excellent environment to study as a doctoral student in his laboratory. All of my productive researches are credited to none other than him. His advanced perspective and thoughtful advises led me to successful achievements.

I would like to demonstrate my gratitude to Associate Professor Tian Song in Tokushima University for giving so much help from both research and daily life.

My appreciation also goes to Professor Masaharu Imai, and Associate Professor Masanori Hashimoto in Osaka University for detailed review and insightful suggestions.

I am deeply grateful to Professor Tatsuhiro Tsuchiya, Professor Koji Nakamae, Professor Haruo Takemura, Professor Makoto Nakamura, Associate Professor Yoshinori Takeuchi, Associate Professor Nakagawa Hiroyuki, Associate Professor Katsuyoshi Miura, Associate Professor Yoshihiro Kuroda, and Associate Professor Kiyoshi Kiyokawa in Osaka University for useful advices.

I would like to express my sincere appreciation to Assistant Professor Masahide Hatanaka, and Assistant Professor Constantin Siriteanu in Osaka University for technical and other supports in laboratory.

I also appreciate Professor Isao Shirakawa, Mr. Shozo Tsukahara, Mr. Hiroyuki Okuhata, Dr. Masanao Ise and Dr. Kenji Watanabe in Synthesis Corporation for providing me a chance to engage on jobs with regard to VLSI design.

I would like to thank Dr. Takehiko Amaki, Dr. Igors Homjakovs, Dr. Masashi Okada, Dr. Dawood Alnajjar, Dr. Jaehoon Yu, Dr. Hiroaki Konoura, Dr. Kosuke Nakajima, and Dr. Ryo Harada for fruitful discussions and advices.

I would like to thank my closest colleagues Ms. Twe Ta Oo and Mr. Younghun Ko for helpful discussions.

I express my appreciation to all the members of Onoye Lab in Osaka University for all kinds of supports in laboratory.

I would like to thank Mrs. Mayuko Nakamura, Ms. Tomomi Kondo and Ms. Yuki Yoshida for their various supports throughout my student life.

I would like to express my sincere appreciation to all the members of Osaka Chinese Presbyterian Church who have accompanied me during these three years. I also would like to thank all of my friends for having good times.

I want to give my sincerely thanks to my family, my father and my mother, thanks for your always supporting in my study. Without you, it would be impossible for me to gain these achievements. I wish you all a healthy body.

I owe my most sincere gratitude to my wife and her parents for everything. Thanks GOD!

Contents

List of tables

List of figures

Chapter 1

Introduction

This chapter describes the background and the objectives of this thesis. This thesis focuses on the fast mode decision algorithms for the H.265/HEVC. HEVC has incorporated a series of technologies and algorithms. These algorithms help HEVC to achieve a high compression efficiency. However, these technologies also increase the complexity. In order to find the best encoding mode for a block, a huge number of combinations of block sizes and modes will be checked exhaustively. Moreover, larger sized transform is a newly-added feature in HEVC and the traditional design of this part cannot be directly reused. Also, the transform of HEVC involves heavy computations and the hardware allocation for this part should be considered carefully. Therefore, in this work, a course of fast mode decision algorithms and its corresponding hardware architectures as well as the hardware design of the transform of H.265/HEVC are proposed.

1.1 Background and motivation

For the past several years, the great achievement in video compression area is the creation of the H.264/MPEG-4 advanced video coding (AVC) standard [1], which has been world-widely used in a variety of applications.

As technology advances, the definition and the quality of the digital video materials have improved fast and steadily. More and more electronic terminals become capable to support high resolution video capture and display. Therefore, studies on performance improvement of video codec are becoming common concerns. The standardization of the next generation video coding standard, i.e. high efficiency video coding (HEVC), was formally launched by the same video compression standardization organizations, named ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG), in January 2010, and the first edition of the H.265/HEVC standard [2] was finalized in January 2013.

As a successor, the design of HEVC codec [3] has incorporated state-of-the-art technologies and algorithms, as shown in Figure 1.1. Following features are contained: intra prediction with 35 modes, inter prediction with merge technique, large-sized transform coding, high-efficiency entropy coding, de-blocking and sample adaptive offset (SAO) filters.

These features help HEVC to achieve significantly high compression efficiency. However, these features also increase the computational complexity. As is depicted in Figure 1.2, the blocks with solid lines denote the mode decision process in HEVC. In this process, first, the size of a block is decided according to its depth. Second, the inter and intra prediction modes are checked to find the best one for this block. Third, this block is split into the next depth

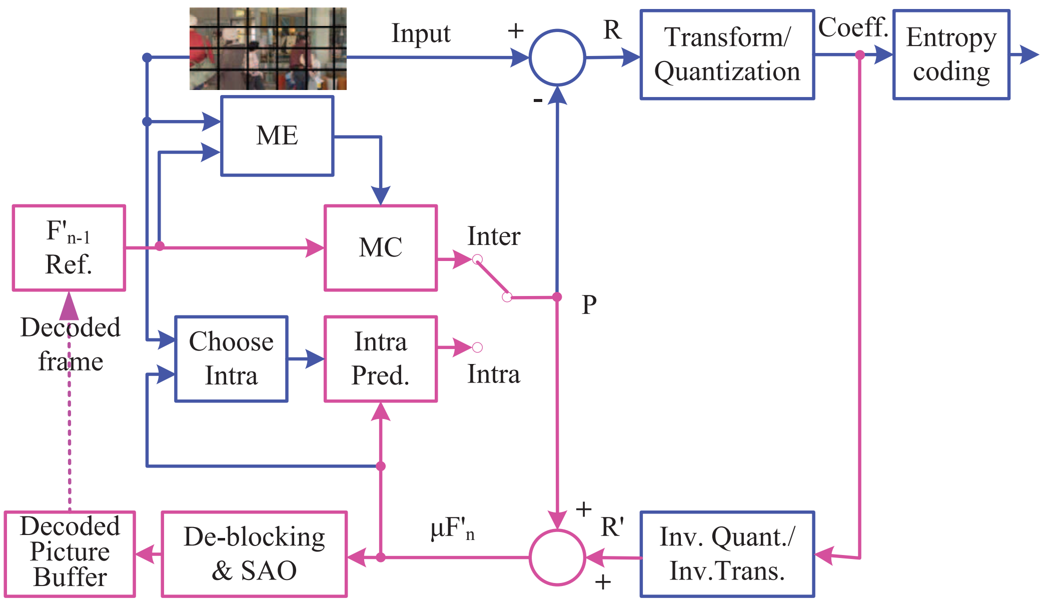

Figure 1.1: Simplified block diagram of HM encoder.

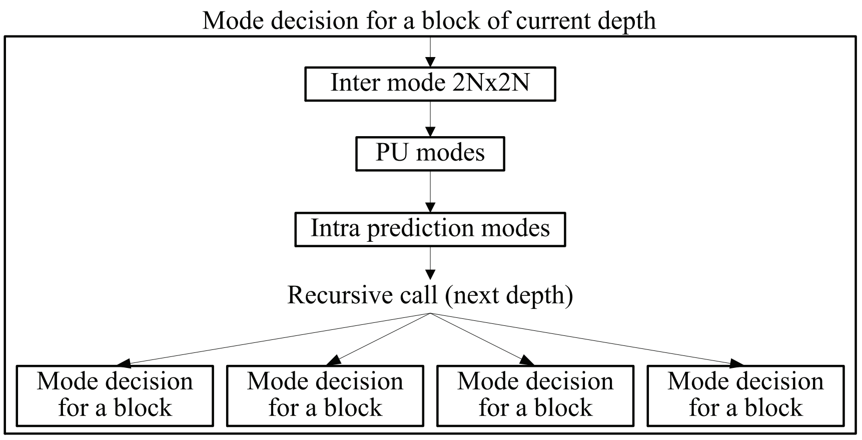
Mode decision for a block of current depth
Inter mode 2Nx2N
PU modes
Intra prediction modes
Recursive call (next depth)
Mode decision for a block
Mode decision for a block
Mode decision for a block
Mode decision for a block

Figure 1.2: The mode decision process for a block.

and the first two steps are repeated to find the best modes for the blocks in the next depth. Finally, these two depths are compared to determine the better depth and mode.

On the basis of the above analysis, in order to find the best encoding mode for a certain block, a huge number of combinations of block sizes and candidate modes have to be checked, which is very time-consuming.

1.2 Previous researches

As stated, the complexity of HEVC has increased a lot, compared with the previous standards. In references [4–7], a series of simulations of complexity analysis have been conducted. According to the results of complexity analysis of HEVC reference encoder [4, 5],

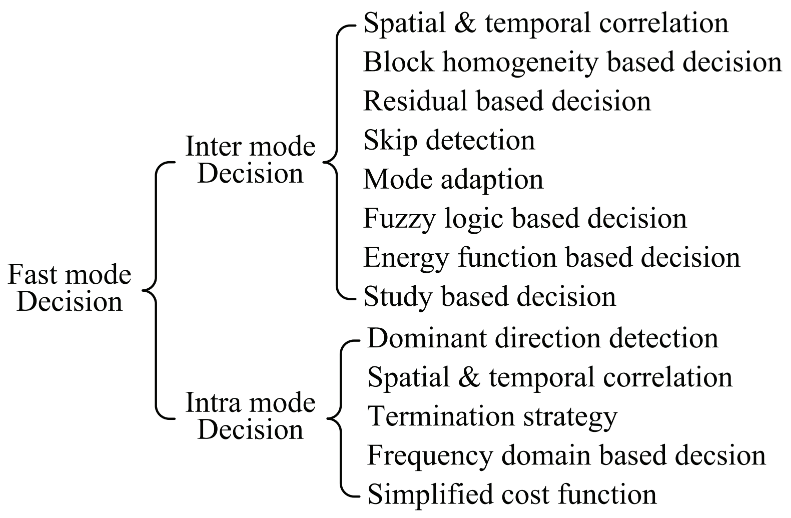
Fast mode Decision
Inter mode Decision
Spatial & temporal correlation
Block homogeneity based decision
Residual based decision
Skip detection
Mode adaption
Fuzzy logic based decision
Energy function based decision
Study based decision
Intra mode Decision
Dominant direction detection
Spatial & temporal correlation
Termination strategy
Frequency domain based decsion
Simplified cost function

Figure 1.3: Classification tree of proposals on fast mode decision algorithms for H.264/AVC.

the rate_distortion optimization (RDO) [8, 9] based mode decision encoding stage always holds a high percentage of the encoding time.

Therefore, a lot of research works have been proposed to reduce the computational complexity caused by the exhaustive mode decision process adopted in the encoder of HEVC.

1.2.1 Fast mode decision algorithms for H.264/AVC

Since both the newly H.265/HEVC and the previous H.264/AVC adopt the well-known hybrid block- based coding framework, most of the fast mode decision algorithms proposed for H.265/HEVC can trace back to the fast mode decision researches for H.264/AVC. Hence, we first make a summary of the fast mode decision algorithms for H.264/AVC. Since the beginning of the standardization of H.264/AVC, how to make a fast and accurate mode prediction has been an important research topic and lots of research works have been proposed. These previous algorithms are categorized according to decision stage and criteria, as shown in the classification tree in Figure 1.3.

There are roughly two categories of algorithms: fast inter mode decision and fast intra mode decision algorithms. References [10–16] propose to reduce the computational complexity of the variable block-size motion estimation based on the temporal and spatial correlation in a video sequence. C. Duanmu *et al.* [10] present an early detection of the 16x16 block or SKIP mode according to the condition that the sum of absolute difference corresponding to the (0,0) motion vector is less than a predetermined threshold, and one of the chosen modes of the spatially neighboring is SKIP or 16x16. K. Chang *et al.* [11] use the best modes of the spatial neighboring blocks and the most correlated block from the previous frame as prediction candidates. In reference [12], the mode decision process can be terminated by referring the lowest RD_Cost among the co-located blocks of the previous encoded frames. Authors of [13] propose to classify the blocks into three sub-sets on the basis of the luminance difference between current block and its collocated block in previous frame, and then check some certain modes for each sub-set. References [14, 15] propose novel intra prediction mode selection schemes in P-frames by using the temporal correlation between the intra predicted block in the current frame and the corresponding block in its reference frame. B. Hilmi *et al.* [16] propose reducing the number of candidate modes using direct information of the co-located blocks from previous frames.

Authors of references [17, 18] propose using homogeneous character of a block to avoid splitting into smaller blocks. In [17], the homogeneity of the block is evaluated based on the edge information calculated by Sobel operator, while in [18], a fast hierarchal cross differences algorithm is proposed to estimate the homogeneity of the block.

References [19–21] propose some fast inter mode selection algorithms according to the homogeneity of the residual block. A. Yu *et al.* [19] evaluate the homogeneity of the residual block by comparing the difference between current and co-located residual blocks. In [20,21], the homogeneity of each portion of the current residual block is used to decide whether this block needs to be split into smaller blocks.

In [22], I. Choi *et al.* propose a fast SKIP mode detection algorithm on the basis of block size, reference frame, motion vector, and the transformed coefficients. Authors of [23] propose to construct a priority-based mode candidate list by adaptively projecting the modes as points onto a 2-D map, and perform mode decision of the modes from this list with early termination conditions.

Reference [24] presents a mode classification strategy that fuzzy logic technique is used to determine the possible candidate modes for a certain block. On the other hand, in work [25], it is proposed that the energy function of the transformed coefficients is adopted to sort the blocks to certain categories. Moreover, a learning based algorithm is proposed in [26], where the number of candidate inter modes is reduced according to the statistical data obtained from the previously encoded frames.

For the fast intra mode decision algorithms, in references [27–30], some fast algorithms are proposed based on the local edge information of the block, e.g. dominant direction. F. Pan *et al.* [27] advocate using Sobel operator to create a local edge direction histogram and a small part of intra prediction modes are chosen for RDO calculation based on the distribution of this histogram. In [28], the dominant direction is estimated based on the sum of absolute error of the sub-sampled block. C. Miao *et al.* [29] propose an extensive pixel-based algorithm that the direction strength of a certain mode i is calculated based on the differences between every two neighboring pixels located along the direction defined by the mode i. In reference [30], the same edge detection strategy is proposed and variance of current block is calculated to decide whether smaller blocks should be checked or not.

References [31–33] exploit the correlation between optimal coding modes of temporal and spatial adjacent blocks to reduce the computational complexity of the encoding. J. Xin *et al.* [31] propose to measure the difference between current block and its co-located block in the previous frame. If they are close enough, the current block will reuse the mode of its co-located block and the entire mode decision process is skipped. Authors of [32, 33] present a method based on the statistical properties (e.g. mode information and block size) of references pixels and adjacent blocks.

Some early termination strategies are given in [34–36]. An early termination method with adaptive threshold is developed by H. Zeng *et al.* [34], and the candidate modes are selected according to their Hadamard distances and prediction directions. The authors of [35] exploit the correlation between the sum of absolute transform difference and rate distortion to terminate or skip some unlikely prediction modes efficiently. Reference [36] proposes to construct a priority-based mode candidate list, and perform mode decision of the modes from this list

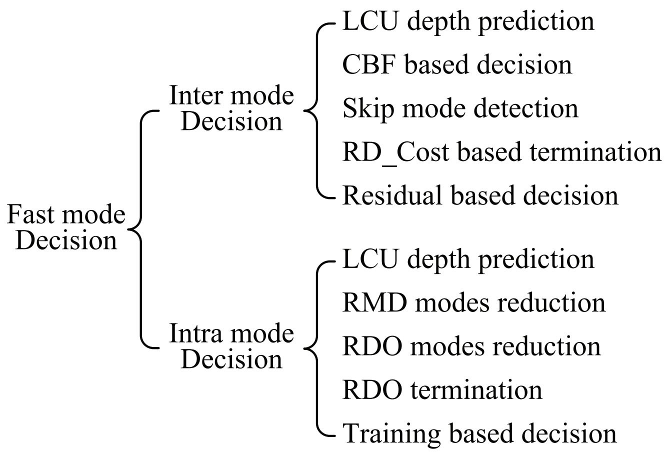

Figure 1.4: Classification tree of proposals on fast mode decision algorithms for H.265/HEVC.

with early termination conditions.

In reference [37], the proposed algorithm finds the candidate modes by analyzing the transformed coefficients of its neighboring blocks. This method modifies the mode decision scheme, which estimates the direction of current block in frequency domain. C. Tseng *et al.* [38] propose a simplified rate-distortion cost function to reduce the calculation complexity.

1.2.2 Fast mode decision algorithms for H.265/HEVC

On the basis of the fast mode decision algorithms proposed for H.264/AVC, a lot of researches have been conducted to accelerate the mode decision process for H.265/HEVC. These previous algorithms are also classified according to decision stage and criteria, as shown in the classification tree in Figure 1.4.

There are also roughly two categories of algorithms: fast inter mode decision and fast intra mode decision algorithms. The large coding unit (LCU) depth prediction schemes of both categories deliver a suitable prediction of the depth of current block. References [39–43] propose using the depth information of LCUs from reference frames and/or neighboring LCUs to predict the suitable depth range for current LCU, in order to skip unnecessary depths and/or terminate current depth check. S. Tai *et al.* [44] suggest that some specific depths of a coding unit (CU) quadtree can be eliminated by referring to the coding information of the co-located CUs and CUs adjacent to the co-located CUs. M. Cassa *et al.* [45] present a Top Skip technique in which the larger CU sizes are avoided by selecting a starting LCU depth based on an observation that there exists a high correlation between the minimum depth of the current LCU and that of the co-located LCU in the previous frame. In reference [46], proposes to skip some unlikely depth based on the texture complexity of the down-sampled LCU. Authors of [47] propose an algorithm that collects relevant and computational-friendly features to assist decision of the depth of LCU on the basis of a predefined Bayesian decision rule.

The algorithms of the coded block flag (CBF) based decision are designed to seek an early detection of the SKIP mode. In references [48, 49], the difference motion vector (DMV) and

CBF after searching the mode 2Nx2N are simply checked to detect the SKIP mode. Reference [50] introduces a method to reduce the encoding complexity by simply investigating the CBF. For the SKIP mode decision scheme, the authors of [51] propose a simple tree-pruning algorithm to exploit the observation that sub-tree computations can be skipped if the coding mode of current block is SKIP. All the mentioned proposals from the above two sub-categories have been adopted in the HEVC test model HM10 [3]. Moreover, the authors of references [52, 53] present a depth range selection mechanism according to the location of a depth with SKIP mode.

As for the algorithms of RD_Cost based termination, some proposals advocate terminating the mode decision process by comparing the RD_Cost of current CU with pre-defined thresholds. Reference [54] presents a strategy that if the current best RD_Cost is less than a threshold, the mode decision process will stop. Authors of [55, 56] investigate a strategy that no further splitting process is necessary when the RD_Cost of current CU is lower than the cost obtained from the modes already coded with SKIP mode. Work [57] proposes to terminate the estimation of all the RD_Costs of the merge candidates when for any candidate of the list, the cost of the SKIP mode is inferior to the minimum of any previous RD_Costs already computed for previous candidates.

For the residual based decision proposals, authors of [58] propose terminating the splitting process of LCU if the average and variance of the best residual of current CU are less than a pre-defined threshold. The authors of reference [59] design a fast algorithm for residual quadtree coding that replaces the original depth-first residual transform process by a merge-and-split transform process. K. Choi *et al.* [60] propose to prune a residual quadtree in the early stage based on the number of nonzero transformed coefficients. P. Chiang *et al.* [61] investigate to reduce the computational complexity of residual coding according to a fast zero block detection scheme based on the sum of absolute difference value which is available in the inter prediction computation.

In HEVC, the number of modes for intra prediction has been increased to 35, which will lead to a higher computational complexity when exhaustively checking over all candidate modes. Therefore, intra mode decision algorithms have been proposed to reduce the number of modes to be checked. Some authors propose reducing the number of modes checked by the rough mode decision (RMD) process. A. Motra *et al.* [62] propose to use the direction information of the co-located block of a previous frame along with the neighboring blocks of current frame as candidates to reduce the number of intra prediction modes in the RMD process. The work in [63] presents an intra mode decision algorithm that reduces the complexity by taking into account the dominant edge orientation of the current and previous depths and then calculating the dominant edge based on the differences between neighboring sub-sampled pixels along a certain direction. W. Jiang *et al.* [64] use the Sobel edge operators to detect the gradient of a block to accelerate the intra prediction process in HEVC, while in reference [65], the gradient of the block is calculated by summing the absolute differences between neighboring pixels located along some pre-defined directions. G. Chen *et al.* [66] propose a fix-point arithmetic based edge detector to analyze the textures of the source image block and select a small set of possible modes to send to the RMD process.

Algorithms that reduce the number of modes checked by the RDO process have also been proposed. Reference [67] proposes to use the mode information from spatial neighbors to reduce the number of candidates checked by the RDO process, ensuring that the most probable mode (MPM) is always checked in the RDO. This method had been added into the HM [3].

Reference [68] proposes a fast intra prediction mode decision based on estimating the rate distortion cost using Hadamard transform to reduce the number of modes sent to the RDO process. References [69, 70] propose analyzing the statistics generated by the RMD process to select a smaller set of modes output from RMD process and then send it to the RDO process combined with DC mode and MPM. The authors of [71, 72] propose to determine the best prediction mode of a CU by referring to the mode of a parent CU in the previous depth. In the technique in reference [73], a mode output from RMD can be selected as the best mode when it is similar to modes from a co-located CU in a previous frame and spatial CUs in the current fame. Various RDO termination strategies have been presented in [74] and [75].

In the last sub-category, another group of studies have been conducted based on various training methods [76–79]. Information (such as threshold) obtained from the study process can be used to stop the mode decision process of current CU if necessary or to skip modes that are thought to be unlikely.

1.2.3 Hardware architecture of the transform for H.265/HEVC

Same as the previous video coding standard, an encoder can be decomposed into a series of coding stages, and each stage is conducted on the basis of different compression norm. Among these coding stages, transform coding stage plays a relatively important role. Generally, transform coding is achieved by discrete cosine transform (DCT), and the purpose of this stage is to concentrate the energy of a residual block generated from the prediction stage to the first few numerical coefficients, enabling the following quantization and entropy coding stages to be performed more efficiently.

In order to decrease the computational complexity and solve the mismatch problem between forward and inverse transforms, integer DCT rather than floating DCT is used. The largest transform size adopted in H.264/AVC is 8x8, while in HEVC, even larger sized DCT including 16x16 and 32x32 is provided, since larger sized transform could achieve higher compression ratio.

The theory of the transform for HEVC is proposed in [80–83]. The transform size is ranging from 4x4 to 32x32, and the computational complexity of which is much higher than that of H.264/AVC (transform size is 4x4 or 8x8). The corresponding transform matrixes are also given in [80–83]. The transform of HEVC is implemented on the basis of transform unit (TU) in the HM [3] using butterfly combined with multipliers method. However, the matrix multiplication is inevitable based on this method. Hence, in reference [84], it is proposed to decompose the transform matrixes into orthogonal matrixes and general matrixes with smaller elements. The number of different elements in these matrixes can be reduced after decomposing, so that the number of multiplication can be reduced.

Since the development of H.264/AVC, a variety of researches [85–90] have been proposed about the hardware implementation of its transform. Authors of [85–87] propose a direct 2-D forward transform architecture according to the symmetry properties of the transform matrixes, while taking account of eliminating drift effects, multiplying free, and reducing memory bandwidth. In [88, 89], a separate 1-D forward transform architecture with a transposition memory, realized by parallel register array, are proposed. K. Chen *et al.* [90] present a high-performance direct 2-D transform coding IP.

However, larger sized transforms used in HEVC increase the computational complexity in two aspects. One aspect is the computation logic of the transform. For larger sized transform logics need more multipliers and adders, so that more hardware resource will be consumed.

The other problem is the cost of the transpose memory. In order to store more intermediate results, more areas will be consumed by the transpose architecture. To solve the above two problems, several latest researches [91–94] about the HEVC transform have been reported.

The authors of reference [91] present area- and power-efficient architectures for the transform of HEVC. The proposed structures could be reusable for DCT of all sizes, and power-efficient structures for folded and full-parallel implementations of 2-D DCT are proposed. R. Jeske *et al.* [92] provide a 16-point 1-D transform architecture used by a 16x16 2-D transform, and this design is conducted in a fully combinational way. In reference [93], a fully pipeline based 2-D transform engine supporting variable block sizes with the efficient hardware utilization is proposed. A unified architecture for inverse transform and forward transform is devised through the algorithm optimization. S. Park *et al.* [94] present high throughput and power-efficient architectures with efficient matrix multiplication schemes.

Moreover, some works [95–97] propose combined transform architectures that supporting several popular video coding standards. In references [95, 96], unified transform architectures are investigated, which can support both the existing video coding standards like H.264/AVC [1], MPEG-2/4 [98, 99], AVS [100], VC-1 [101], and the HEVC [2]. M. Martuza *et al.* [97] present a shared architecture which can compute the 8x8 inverse transform of the HEVC from that of the H.264/AVC using a new mapping technique.

1.3 Objective and outline of this thesis

For the LCU depth prediction algorithm proposed in previous works, the depth information of a co-located LCU from a reference frame is used to make a prediction. Hence, the depth information of all possible reference frames needs to be stored, and this of course results in the consumption of a large amount of memory. Therefore, in this dissertation, we propose using the co-located LCU from the previous frame in an encoding order. This means that the depth information of only one frame needs to be saved while still maintaining a better performance. In the residual based decision algorithm mentioned above, the residual information is only used to terminate the current depth mode decision, but in our proposal, this obtained information is used not only to terminate but also to skip over unlikely modes and directly split current CU into the next depth. We also propose a remedy strategy to prevent prediction error propagation. For the RMD modes reduction algorithms for intra prediction, such as those using the Sobel operator, the algorithm itself is very complex, and thus we also propose a low complexity hardware-oriented fast intra prediction algorithm. Hardware synthesis results show that only 1/9 of the resources are consumed compared with previous works.

The purpose of this work is to reduce the computational complexity of the HEVC encoder without causing any noticeable performance degradation. First, in order to determine the size of current block, the depth information of a co-located block from the previous frame is used to predict the proper block size. Then, for a certain sized block, the residual generated by inter prediction is analyzed to determine whether to terminate current check or to skip over unnecessary modes and split into smaller sizes. Finally, after inter prediction, a hardware-oriented low complexity fast intra prediction algorithm is proposed. Our algorithm adopts a fast discrete cross difference (DCD) to detect the dominant direction of the coding unit. Based on DCD information, only a subset of the 35 candidate modes are selected for the RMD process. We also propose four simple but efficient early termination strategies for

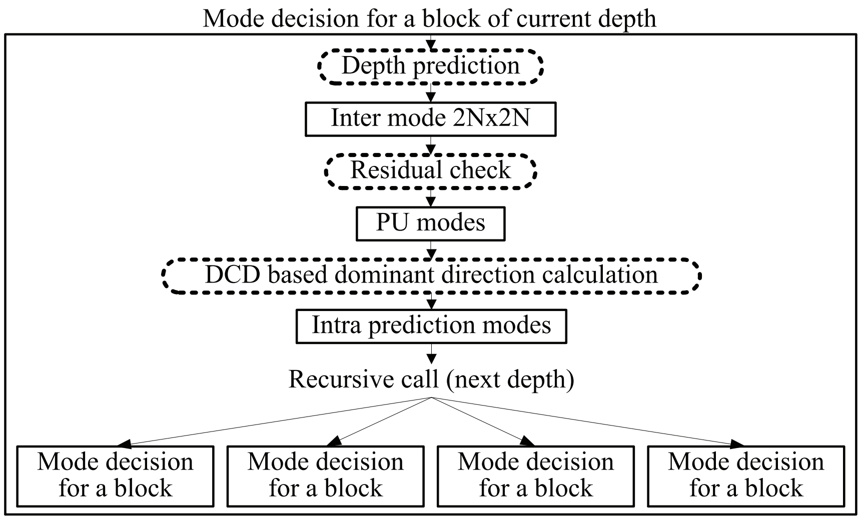
Mode decision for a block of current depth
Depth prediction
Inter mode 2Nx2N
Residual check
PU modes
DCD based dominant direction calculation
Intra prediction modes
Recursive call (next depth)
Mode decision for a block
Mode decision for a block
Mode decision for a block
Mode decision for a block

Figure 1.5: The proposed mode decision process for a block.

terminating the RDO process properly. As shown in Figure 1.5, the blocks marked by dotted lines are the added processes we propose in this work.

The rest of this thesis is organized as follows. Chapter 2 gives a brief introduction about the H.265/HEVC, including the quadtree structure based block partition scheme, advanced prediction and transform coding tools, improved in-loop reconstruction filters, and three prediction structures. Chapter 3 proposes a course of low complexity fast mode decision algorithms, namely depth prediction, residual check, and fast intra mode decision algorithm. The performances of individual or combined algorithms are also analyzed. Chapter 4 presents the corresponding hardware architectures of the proposed fast mode decision algorithms in Chapter 3, and the synthesis results of each architecture are also provided. Chapter 5 describes a low cost and high throughput transform architecture for 1-D and 2-D transform in HEVC. This architecture supports a transform size ranging from 4x4 to 32x32. Chapter 6 makes a conclusion about this thesis.

Chapter 2

High efficiency video coding

2.1 Introduction

High efficiency video coding (HEVC) is the video coding standard of the VCEG and MPEG. The main goal of the HEVC standardization effort is to enable significantly improved compression performance relative to existing standards – in the range of 50% bit-rate reduction for equal perceptual video quality.

The HEVC standard adopts the well-known hybrid block-based coding framework, containing advanced intra prediction with 35 modes, advanced motion-compensation prediction with merge technique and large-sized transform coding followed by high-efficiency entropy coding. Moreover, the reconstructed pixels will be processed by de-blocking and SAO filters before sent to the decoded picture buffer. For special case, the lossless coding [102] is also supported in the HEVC.

In the following sections, a brief introduction about the key technologies of HEVC is presented, and more detail information about HEVC is provided in references [103, 104].

2.2 Block partition structure and parallel scalability

2.2.1 Block partition structure

Other than the previous video coding standards, this HEVC uses a flexible quadtree block partition structure [105, 106] that enables effective use of different block sizes during the prediction and transform coding processes. As depicted in Figure 1.1, an input video image is first divided into a sequence of LCUs. An LCU, comprised of a 2Nx2N block of luma samples, can be part of a slice or a tile defined by the configuration file. LCU is broadly analogous to the concept of macroblock in H.264/AVC and an index is allocated to each LCU in raster-scanning order. All encoding procedures are conducted on the basis of LCU. Each LCU can be recursively split into four smaller blocks with the same size, called CUs. One possible way to split an LCU into CUs is illustrated in Figure 2.1.

Each CU located inside an LCU is indexed in a Z-scanning order. The CU is the basic coding region used in the intra/inter prediction coding procedure. The shape of an CU is always square and may take a block size starting from the size of the LCU going all the way down to a minimum size of 8x8 luma samples. This recursive CU splitting process can generate a content-adaptive quadtree structure, providing a flexible partition scheme and enabling efficient use of large and multiple block sizes.

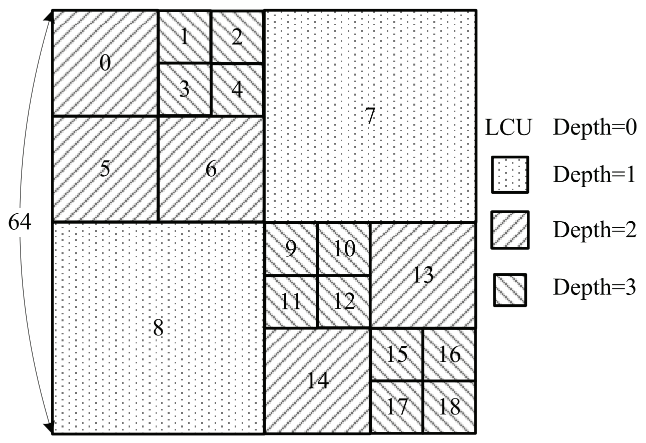

Figure 2.1: Example of CU partition scheme.

As shown in Figure 2.1, different sized CUs are given different depth indexes. For example, the 1st CU with a size of 16x16 is indexed by depth 2, while the depth of 2nd CU sized by 8x8 is 3, and the whole LCU with a size of 64x64 is indexed by depth 0. Therefore, the size of CU is expressed in terms of depth. There are totally four depths from 0 to 3 defined in HEVC.

2.2.2 Parallel scalability

For the sake of overcoming the limitations of the parallelization approaches employed in H.264/AVC, two strategies aiming at enhancing the parallel scalability [107] of the HEVC, have been included in the HEVC, namely Tiles and wavefront parallel processing (WPP). Both of these two approaches propose to divide each frame of the video sequence into multiple partitions that can be processed in parallel. An integer number of LCUs are contained in each partition.

When the Tiles strategy is enabled during encoding process, a picture will be divided into rectangular groups of LCUs separated by vertical and horizontal boundaries, as shown in Figure 2.2. The thick lines in this figure represent the Tile boundaries. The number of tiles and the location of the corresponding boundaries can be controlled by the configuration command. The scanning order of each LCU will be also changed according to the Tiles, where LCUs will be scanned from tile to tile. An example is given in Figure 2.2. To facilitate high level parallel processing, each tile can be processed independently.

When the WPP strategy is used in the codec, each LCU row of a frame is recognized as a separated partition, and can be processed independently. An example of the WPP approach is illustrated in Figure 2.3. In this example, the LCU rows indexed by 1, 2, 3, 4 can be encoded in parallel by the corresponding threads 1, 2, 3, 4. So that, several LCU rows can be encoded in parallel. When the encoding of the LCU rows 1, 2, 3 finishes, that of the LCU rows 5, 6, 7 starts in parallel. Additionally, in order to further reduce the coding losses, the coding information, e.g. probabilities, from the second LCU of the previous LCU row, can be taken as reference when encoding current LCU row. For example, when thread 2 starts to encode LCU 8 in LCU row 2, the coding information of LCU 1 will be used as reference. So, each thread starts only when the first two LCUs are encoded in the previous thread.

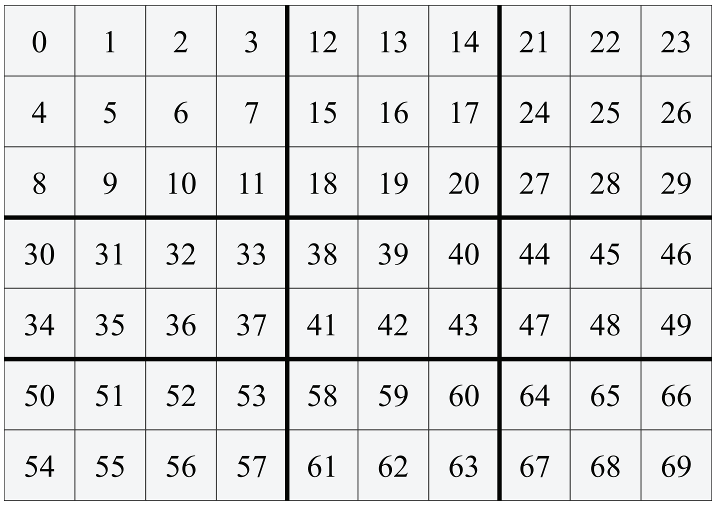

Figure 2.2: Example of dividing a frame into nine Tiles.

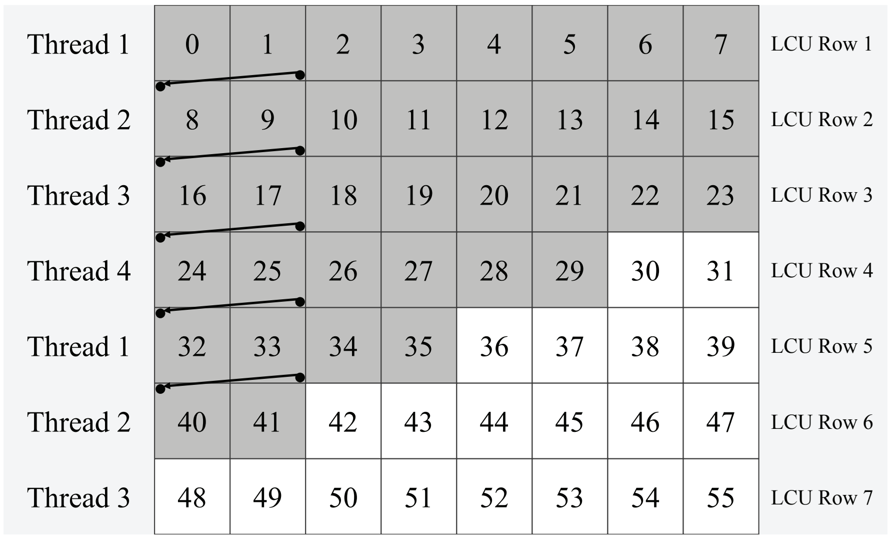
Thread 1 0 1 2 3 4 5 6 7 LCU Row 1
Thread 2 8 9 10 11 12 13 14 15 LCU Row 2
Thread 3 16 17 18 19 20 21 22 23 LCU Row 3
Thread 4 24 25 26 27 28 29 30 31 LCU Row 4
Thread 1 32 33 34 35 36 37 38 39 LCU Row 5
Thread 2 40 41 42 43 44 45 46 47 LCU Row 6
Thread 3 48 49 50 51 52 53 54 55 LCU Row 7

Figure 2.3: Parallel processing of each LCU row in WPP.

2.3 HEVC video coding technology

2.3.1 Prediction coding

As is stated, there are 35 intra prediction candidate modes [108] for each CU for all permissible sizes, as shown in Figure 2.4. The mode indexed by 0 is called Intra_Planar [109], which assumes the current block as an amplitude surface with horizontal and vertical slopes derived from the boundaries. The mode 1 is defined as Intra_DC, which is similar as the DC mode defined in H.264/AVC. Other directional modes indexed by 2...34 are Intra_Angular modes.

In practice, the intra prediction process is generally divided into two steps. Firstly, a rough

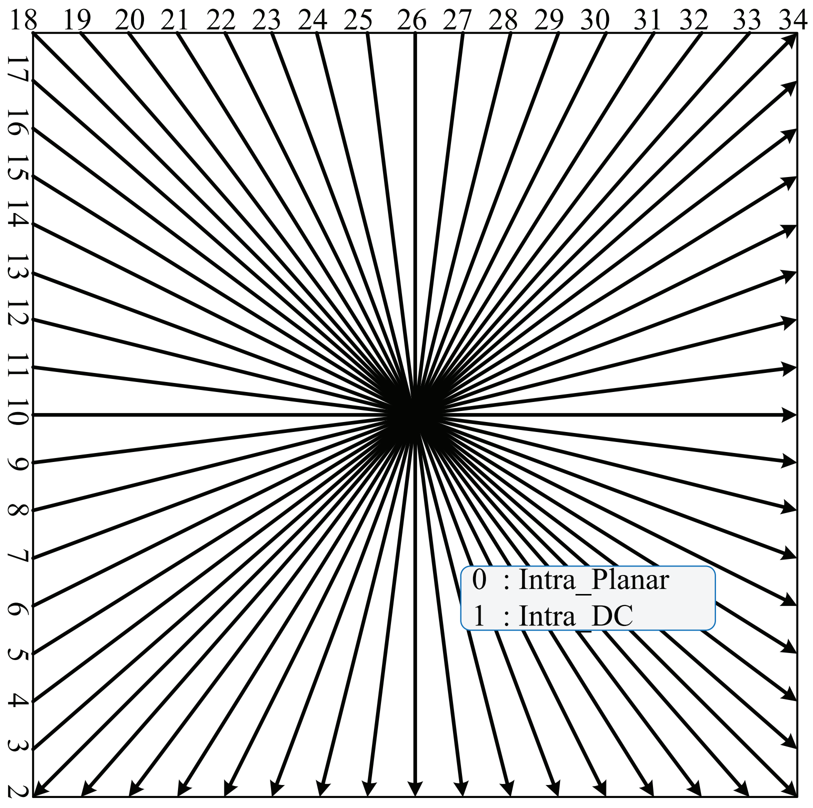

Figure 2.4: Intra prediction mode directions.

mode decision is conducted over the 35 candidate modes to select a number of sub-optimal modes based on the Hadamard transform results and estimated mode bits. Then, the chosen subset of modes are sent to the RDO process one by one to find out the best mode with lowest cost, denoted as RD_Cost, evaluated by the distortion and the encoded mode bits. Based on the two-step search strategy, the complexity can be relatively reduced without decreasing the quality remarkably. However, there still exits a high computational complexity.

For inter prediction, in order to match the boundary of real objects, a CU can be go on divided into smaller blocks, called prediction units (PUs), as depicted in Figure 2.5. In general, the PU is not restricted to be square in shape, and can be divided into asymmetric rectangle blocks. Basically, all the possible PU partition modes should be checked to find the best one for current CU. Even though a high degree of adaptability is achieved by this PU partitioning scheme, this strategy has certain intrinsic drawbacks, that is, it may result in redundant sets of motion parameters being encoded and transmitted. Hence, a block merging technique [110] has been included in the HEVC.

2.3.2 Transform coding

The transform of HEVC is implemented on the basis of transform unit (TU), which is also the basic unit for quantization. TUs are also formed in a quadtree structure with a root located in residual CU, which means that a residual CU can be recursively divided into four blocks with a size starting from that of residual CU down to a certain size, which is derived from

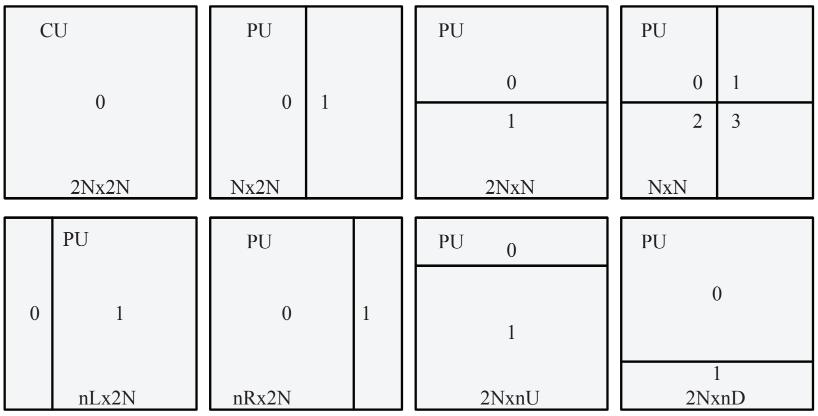

Figure 2.5: Partition modes for inter PU.

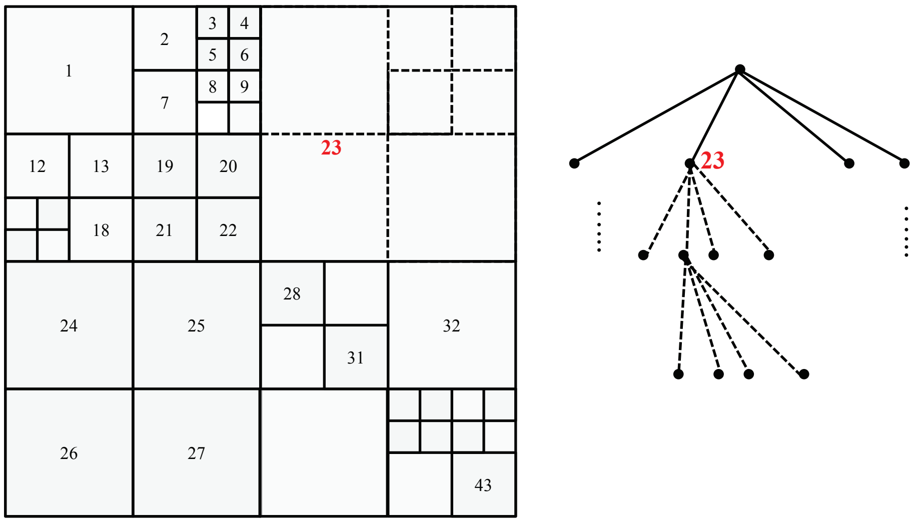

Figure 2.6: Example of TU splitting structure.

the parameter – maximum quadtree depth – specified in the slice header syntax. In HEVC, a TU may take the size from 4x4 up to 32x32 luma samples. Therefore, each residual CU may contain one or more TUs, in which the multiple TUs are arranged in a quadtree structure. As the dashed line illustrated in Figure 2.6, the residual CU indexed by 23 is divided into smaller blocks to transform. This division of residual CU 23 can be converted into tree structure as the quadtree depicted on the right part.

To encode the transformed coefficients, a unique entropy coding tool is used in all configurations of HEVC, i.e. context adaptive binary arithmetic coding (CABAC) [111, 112], compared with H.264/AVC. The core of the CABAC coding engine of HEVC is essentially the same as the CABAC used in H.264/AVC, other than some improvements in the details of practice implementation. For example, relatively fewer contexts are used in HEVC compared with H.264/AVC standard, and also the memory requirements are reduced to benefit both throughput and implementation costs. Moreover, in reference [113], a method for analyzing the dynamic range along the data-path of the residual encoding and reconstruction is presented.

2.3.3 Pixel reconstruction

In HEVC, two filters are designed for the reconstructed pixels, namely de-blocking filter and SAO filter. The de-blocking filter [114] is used to reduce visible artifacts at block boundaries, which are due to the usage of a series of block-based coding tools. The de-blocking process is applied to all samples adjacent to each CU boundary in the same order as the reconstruction process, except following cases when the CU boundary is also the boundary of a frame, or when de-blocking process is not permitted across slice or tiles boundaries. In this process, vertical edges are first filtered by using the horizontal filter, and then horizontal edges are filtered by vertical filter. The minimum CU with a size of 8x8 is the filtering unit, for both luma and chroma components. Hence, the boundaries aligned on a 4x4 block are not filtered, so as to reduce the complexity, which is different from H.264/AVC.

The SAO [115] is a process which modifies the values of the reconstructed pixels after the de-blocking filter through look-up tables. The SAO filter aims to improve the accuracy of the amplitudes of the reconstructed pixels compared with the original pixels. The SAO is applied adaptively to all the pixels, by conditionally adding an offset value to each pixel based on the information from look-up tables defined by the encoder. That is, depending on the local gradient at the position of a certain pixel, a corresponding offset value picked up from a look-up table is added to this reconstructed pixel. The SAO is designed to improve the subjective performance of the codec, and this filter can be selectively turned on or turned off in the configuration file.

2.4 Prediction structure

The HM encoder [3] defines three kinds of temporal prediction structures as test conditions for simulation: intra-only, low-delay, and random access. The management of the reference picture list(s) corresponding to each prediction structure is given in [116, 117].

For the intra-only prediction structure, all the frames of a video sequence are encoded as instantaneous decoding refresh (IDR) pictures. Inter prediction between pictures is not used. Coding parameters for a picture, such as Qp, do not change during the encoding process of the whole sequence, and all the pictures belong to the same layer L_0. Layer is used to describe a group of pictures that are encoded with the same parameters. Layer is also used to denote the importance of this group of pictures. The smaller the index of layer is, the more importance it is. Figure 2.7 gives a graphical illustration of the intra-only prediction structure, where the number assigned to each frame represents the encoding order.

In the test case of low-delay coding, as a normal coding configuration, only the first frame of a video sequence is encoded as an IDR picture. All other pictures are coded as P pictures using inter prediction. These P pictures are classified to different layers, from L_1 to L_3, according to the encoding order. Coding parameters for a frame, such as Qp, will also change as the layer index. Figure 2.8 shows a graphical presentation of this low-delay P configuration. The number associated with each frame represents the encoding order. The arrow in this figure points to the reference picture.

For the random access test condition, a so-called hierarchical B structure is adopted for encoding. Figure 2.9 gives a graphical explanation of a random access prediction structure,

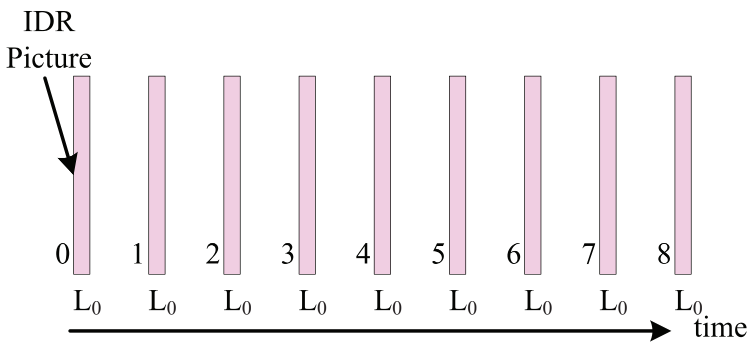

Figure 2.7: Graphical presentation of intra-only prediction structure.

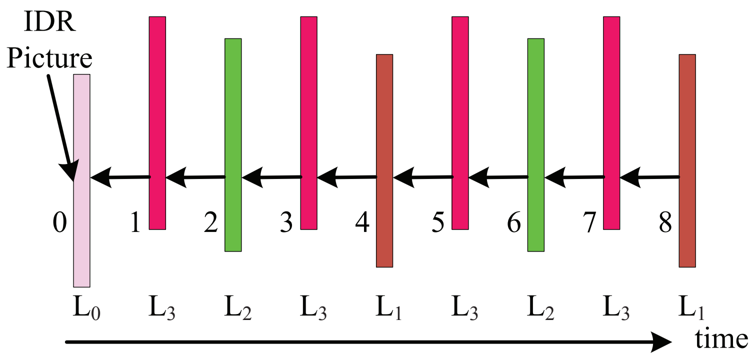

Figure 2.8: Graphical presentation of low-delay P prediction structure.

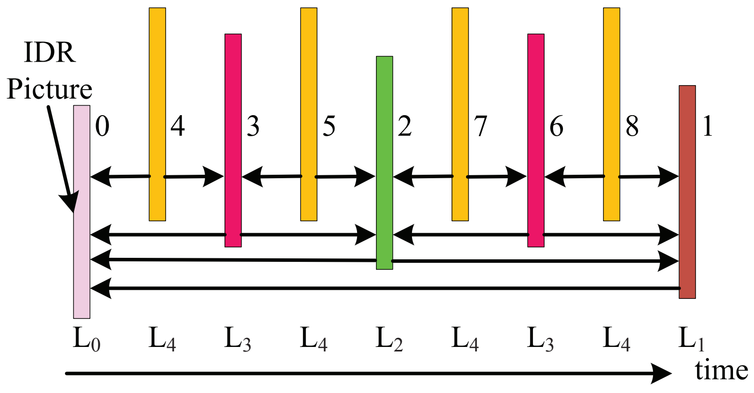

Figure 2.9: Graphical presentation of random access prediction structure.

where the index of each picture denotes the encoding order. Unlike the low-delay coding, an intra picture is encoded at regular intervals defined in the configuration file. Among them, the first intra picture of a video sequence is encoded as an IDR picture while others are non-IDR intra pictures. The remaining pictures located between two intra pictures are encoded as P or B pictures. That is, the picture belong to layer L_1 is a P picture, and others associated with layers from L_2 to L_4, are encoded as B pictures.

2.5 Conclusion

In this chapter, a brief introduction about the H.265/HEVC was presented. In order to achieve a better performance, a course of state-of-the-art techniques had been adopted in HEVC. First, during the prediction and transform coding processes, a flexible quadtree based block partition scheme was supported. Approaches, such as Tiles and WPP, were taken in to enhance the parallel scalability. Then, for the prediction coding, an enhanced intra prediction with 35 candidate modes as well as a flexible motion-compensation prediction with merging technology were employed. Next, larger sized transform and advanced CABAC engine were applied to encode the residual, and a de-blocking filter along with a newly SAO filter were used to filter the reconstructed samples. Finally, three common used test conditions were introduced.

Chapter 3

Hierarchical structure based fast mode decision for H.265/HEVC

3.1 Introduction

In this chapter, a course of low complexity fast mode decision algorithms [118] are presented. First, the depth information of a co-located block from a previous frame is used to predict the size of current block. Next, for a certain sized block, the inter prediction residual is analyzed to determine whether to terminate the current check or to skip over unnecessary modes and split the block into smaller sizes. After inter prediction, a hardware-oriented low complexity fast intra prediction algorithm is proposed. A fast DCD is adopted to detect the dominant direction of the block. In addition, four simple but efficient early termination strategies are proposed to terminate the RDO process properly.

3.2 Depth prediction

It is well-known that, the consecutive frames from nature video sequence tend to be similar, for the objects in the video always moving continuously. As is shown in Figure 3.1, the contents of the two successive frames are almost same, and the partition structure of LCUs located in the same position is also similar to each other. The partition structure is generated using the HEVC reference software HM. Such as the first LCU in both frames, they split in the same way, and generate same sized CUs in the same place.

As stated, the size of CU is expressed in terms of depth, which means that if the depth of current CU can be predicted, its size will also be known. While other impossible sizes (depths) can be directly skipped to reduce the complexity. Based on these analysises, maybe the partition structure of the co-located LCU, which has already been encoded, can be used to predict the depth of current LCU. To achieve this, the relationship between current LCU and co-located one should be made clear first.

3.2.1 Correlation between co-located LCU and current LCU

Generally, an LCU with a larger depth means that there is detail texture inside this block, and hence, smaller CU sizes such as 8x8, are preferred to encode this LCU. Whereas, for LCU within a homogeneous region, large block partition will be better. Therefore, in this work,

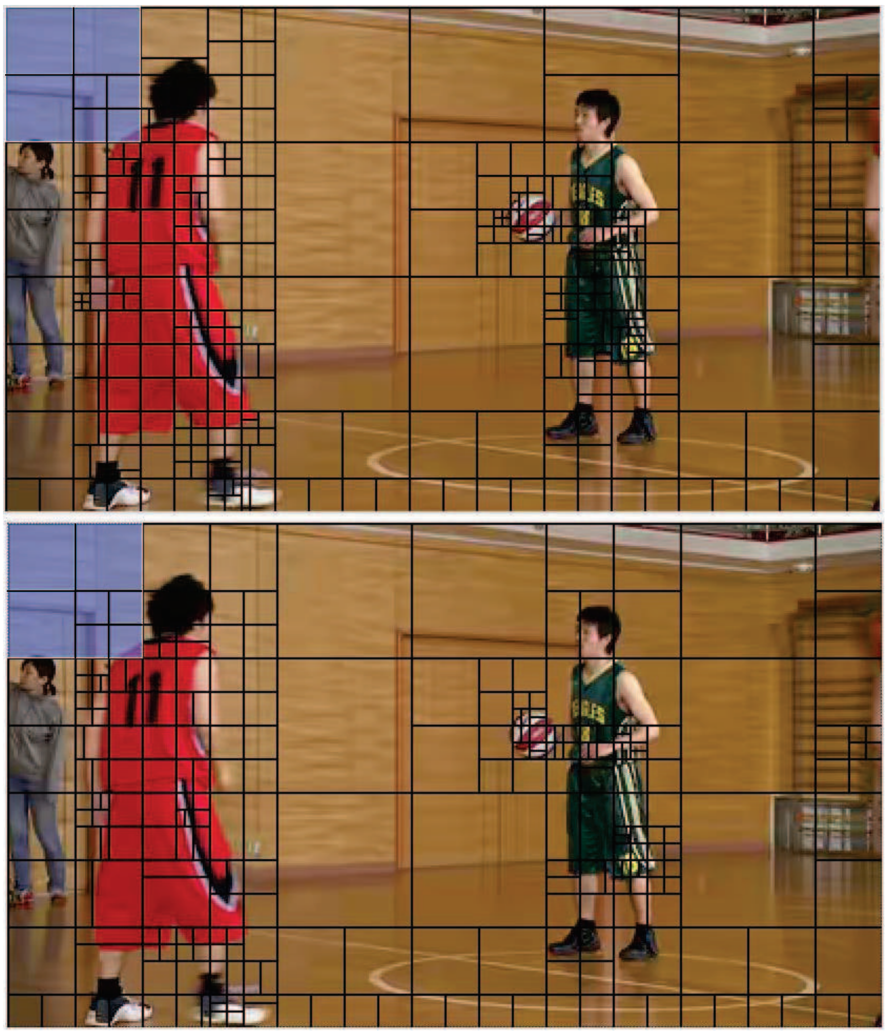

Figure 3.1: Example of LCU partition for two consecutive frames.

we proposed to classify different LCUs into different categories corresponding to different texture characters. There are three categories defined: simple, complex, medial. An LCU will be classified as complex when at least one of the CUs inside this LCU takes the maximum depth, 3 as defined. On the other hand, when all the CUs takes the depth not larger than 1, this LCU will be seen as simple. The rest LCUs will be defined as medial.

The distributions of the occurrence ratios of these three kinds LCUs are estimated over some video sequences, under three different coding structures (Section 2.4), and other configurations are defined in [123]. A single Qp value of 32 is used for this simulation. The results are shown in Table 3.1. From this table, we can see that the distributions of the ratios under different coding structures are different from each other. But for all these three coding structures, the sum of the ratios of simple and complex LCUs always holds a high percentage, while the ratio of medial LCU is around 10%.

Based on Table 3.1, the relationship of the depths of the current LCU and the co-located LCU from the previous frame is analyzed, and the results are given in Table 3.2. A single Qp value of 32 is used for this simulation. In this table, the "NoMaxDepth" column shows the occurrence ratio of such LCU that no CUs inside it takes the maximum depth when the

Table 3.1: Distribution (%) of the ratios of simple, complex, and medial LCUs.

Sequence		All intra			Low-delay P			Random access		
		Simple	Complex	Medial	Simple	Complex	Medial	Simple	Complex	Medial
Class A	Traffic	10.8	81.7	7.5	67.7	21.1	11.2	77.0	15.5	7.5
	PeopleOnStreet	6.1	91.3	2.7	24.4	64.0	11.6	31.4	54.4	14.2
Class B	Kimono	61.9	18.8	19.3	59.4	15.5	25.1	71.7	11.5	16.8
	ParkScene	17.7	78.4	3.9	55.6	34.1	10.3	68.1	24.7	7.2
	Cactus	14.0	77.7	8.3	63.1	25.8	11.1	71.1	20.6	8.3
	BQTerrace	20.6	76.5	2.8	65.8	25.9	8.3	79.5	16.3	4.3
	BasketballDrive	35.2	48.9	15.9	67.8	20.1	12.0	73.5	16.8	9.7
Class C	RaceHorsesC	5.1	90.5	4.3	22.0	63.1	14.9	33.2	50.5	16.4
	BQMall	4.8	90.9	4.3	45.0	45.2	9.8	54.4	34.5	11.1
	PartyScene	0.4	99.2	0.4	31.2	56.0	12.8	48.7	42.2	9.1
	BasketballDrill	1.4	96.0	2.6	52.5	36.2	11.3	58.9	28.9	12.2
Class D	RaceHorses	0.6	98.4	1.0	4.0	81.6	14.4	11.0	64.5	24.5
	BQSquare	0.0	97.7	2.3	26.8	45.2	28.0	52.6	23.6	23.8
	BlowingBubbles	0.1	99.8	0.1	19.8	57.2	23.0	36.5	38.9	24.6
	BasketballPass	11.4	83.0	5.6	42.6	34.4	23.0	50.4	24.8	24.8
Class E	Vidyo1	18.5	61.2	20.3	81.1	5.5	13.4	83.7	4.8	11.5
	Vidyo3	26.8	61.1	12.1	75.8	10.2	14.1	81.1	6.8	12.1
	Vidyo4	23.6	60.3	16.1	79.8	7.3	12.9	82.6	6.0	11.4
AVG		14.4	78.4	7.2	49.1	36.0	14.8	59.2	27.0	13.9

Table 3.2: Correctness (%) of prediction from co-located LCU to current LCU.

Sequence		All intra		Low-delay P		Random access	
		NoMaxDepth	NoMiniDepth	NoMaxDepth	NoMiniDepth	NoMaxDepth	NoMiniDepth
Class A	Traffic	84.1	99.0	90.5	76.4	91.3	68.7
	PeopleOnStreet	86.7	99.5	88.9	98.2	84.2	96.3
Class B	Kimono	97.2	96.4	94.6	91.7	96.3	89.7
	ParkScene	91.9	99.1	84.2	78.4	87.9	76.5
	Cactus	90.2	99.2	93.7	90.3	92.4	85.1
	BQTerrace	92.1	98.6	85.2	68.0	93.0	72.6
	BasketballDrive	91.2	97.7	95.4	90.2	94.6	86.7
Class C	RaceHorsesC	75.9	99.6	76.3	96.7	74.0	94.9
	BQMall	81.0	99.7	87.3	90.5	87.9	89.9
	PartyScene	73.9	100.0	70.1	87.8	82.5	89.2
	BasketballDrill	59.5	99.8	90.2	90.9	88.3	86.6
Class D	RaceHorses	60.0	99.9	49.2	99.8	53.6	97.9
	BQSquare	-	100.0	60.8	85.7	84.3	79.8
	BlowingBubbles	-	99.9	66.8	94.3	77.2	92.0
	BasketballPass	86.2	99.9	92.1	93.8	90.4	90.2
Class E	Vidyo1	96.8	99.5	97.4	58.2	97.0	56.5
	Vidyo3	93.4	98.4	97.1	78.0	96.2	66.0
	Vidyo4	92.0	98.5	97.1	68.2	96.7	64.5
AVG		84.5	99.1	84.3	85.4	87.1	82.4

co-located LCU is classified as simple. The average value of “NoMaxDepth” is around 85%. In other words, if co-located LCU is a simple one, large sized CUs are coded inside it. Hence there will also be a high probability that current LCU will be coded using large sized CUs, and then maximum depth will not appear. While “NoMiniDepth” is equal to the occurrence ratio of such LCU that the minimum depth is not suitable for it when the co-located LCU is tested as complex. The value of “NoMiniDepth” is around 90%.

3.2.2 Complexity reduction

To make a prediction, after encoding one frame, the depth information has to be saved, so that later coded frames can refer to these data. In reference [41], co-located LCUs from reference frames are used, in this case, depth information of all the possible reference frames has to be stored. This will cause a large memory cost, when implemented on hardware platform. In order to reduce the complexity introduced by saving the depth information of co-located LCU, it is proposed in this chapter that co-located LCU from the previous frame in encoding order will be used so that depth information of only one frame needs to be saved.

Based on the definition of LCU classification, to classify one LCU, only the maximum depth of the CUs inside it will be needed. Compared with [39, 42], in which the depth range of current LCU is predicted using minimum and maximum depth information of co-located or spatial neighboring LCUs, in our proposal only the maximum depth information needs to store.

3.2.3 Implementation of depth prediction algorithm

As we can see from Table 3.2, the depth prediction is not always 100% accurate. The optimal depth of an LCU might be skipped due to a wrong prediction, and such prediction errors can accumulate during the encoding process. Therefore, we propose that for the last frame of every N frames, all possible sized CUs corresponding to the whole depth range will be checked in order to maintain the performance. That is, depth prediction is turned off for the last one of every N frames. The number N is selected by taking account of the GOP size and many simulation results.

The proposed depth prediction algorithm is explained by pseudo codes as:

Algorithm Depth Prediction

```
while current LCU != last LCU do
  if (depth information of co-located LCU is available
     && mod(POC of current picture, $N$) != 0 ) then
    for depth = 0 to 3 do
      if (co-located LCU is complex && depth == 0) then
        skip check on current depth;
        depth ++ ;
      else if (co-located LCU is simple && depth == 3) then
        terminate check on current depth;
        break;
      else
        check current depth;
        depth ++;
      end if
    end for
  else
    test all CU sizes according to the whole depth range;
  end if
  store maximum depth information of current LCU;
end while
```

3.3 Residual check

In this section, we will analyze the motion character of each portion of a CU based on its residual information to decide the following encoding flow.

3.3.1 Residual characteristic analysis

After motion estimation, there may be some portions, whose motion compensation residuals may contain a significant amount of energy if motion estimation mismatches these regions. Whereas energy will tend to be lower in matched area. If the mismatched energy is too high, the movement of objects inside this CU is inconformity.

Therefore, a different motion estimation, motion vector to be exact, is needed for the mismatched region. Consequently, current CU will possibly need to split into PUs or smaller sized CUs to match the movement of each small portion.

In order to detect the motion character of each portion inside a CU, we propose to divide the residual block, generated by inter prediction with 2Nx2N partition mode, into sixteen sub-blocks (four, when depth is maximum), as depicted in Figure 3.2. The sizes of these sub-blocks are 16x16, 8x8, 4x4, 4x4 corresponding to the depth 0 to 3, respectively.

Then, the average and sum of absolute difference over average ($SADA$) of each sub residual block are calculated. The average and $SADA$ are calculated by

$$Avg = \frac{1}{S*S}\sum_{m=0}^{S-1}\sum_{n=0}^{S-1} p(m,n), \tag{3.1}$$

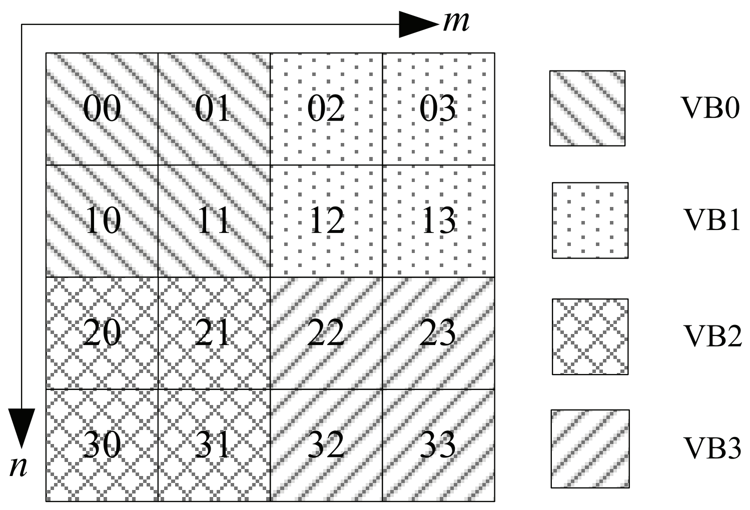

Figure 3.2: Division a CU into 16 sub-blocks.

$$SADA_{ij} = \sum_{m=0}^{S-1}\sum_{n=0}^{S-1} \left|p(m,n) - Avg_{ij}\right|, \tag{3.2}$$

where (i, j) is the index of sub-blocks; $p(m, n)$ is the value of the residual pixel in position (m, n) inside the sub-block indexed by (i, j) and Avg_{ij} is the average of this sub-block. S is the size of current sub-block.

Since $SADA$ is broadly analogous to the concept of variance, it can be used to reflect the character of a block. If the $SADA$ is lower than a threshold, noted as $ThSmooth$, current sub residual block can be recognized as a smooth or homogeneous region.

$ThSmooth$ is defined as

$$ThSmooth = \begin{cases} (d + uiDepth) * ((1 << 2) << 2), & uiDepth = 3 \\ (d + uiDepth) * ((1 << (4 - uiDepth)) << (4 - uiDepth)), & otherwise \end{cases} \tag{3.3}$$

where $uiDepth$ is equal to the index of the current depth; $(1 << (4 - uiDepth)) << (4 - uiDepth)$ stands for the number of residual pixels inside each sub-block. For example, if $uiDepth = 1$ (the size of original residual CU is 32x32, and the size of each sub- block is 8x8), the number of residual pixels inside each of all 16 sub-blocks is 64 ((1 << 3) << 3).

The threshold $ThSmooth$ is selected based on the assumption that if the average difference between every residual pixel and the average of current sub-block is larger than $(d+uiDepth)$, this residual block may be a mismatched portion. Meanwhile, based on this assumption, this threshold can adaptively adjust its value according to the depth of current CU.

3.3.2 CU termination and split conditions

After checking the smoothness of all the sub residual blocks, a termination condition called $p1$ is checked, where the following two terms are tested:

(1) All sixteen sub-blocks are checked as smooth for depth 0, or no more than one sub-block is recognized as unsmooth for other depths.
(2) The absolute values of the averages of all sub-blocks are less than a threshold $ThAvg$.

If $p1$ is satisfied, i.e. both of (1) and (2) are true, the following check over PU partition modes and following depths, will be terminated.

The selection of condition $p1$ is based on the fact that if most of the sub residual blocks are homogeneous, current check is sufficient enough and no more check is needed. If condition $p1$ is not satisfied, the second split condition, named $p3$ is checked, to decide whether current residual block contains several mismatched portions.

Condition $p3$ is evaluated by virtual block (VB) basis, each of which comprises of four neighboring sub-blocks as illustrated in Figure 3.2. A VB is treated as unsmooth in case more than two sub-blocks inside this VB are checked as unsmooth. Following two terms are tested:

(1) There are at least two unsmooth VBs, and current depth is not maximum.
(2) If co-located LCU is simple, which means maximum depth will be skipped, thus current depth should be not equal to 2.

Term (2) aims to make a robust and easy integration of this residual check algorithm and the depth prediction algorithm. According to the depth prediction, if co-located LCU is simple, mode decision process of depth 3 of current LCU will be skipped. If current depth is equal to 2 and $p3$ is decided as true, mode decision process of depth 3 will be conducted. Combine these two situations together, if co-located LCU is simple and current depth is equal to 2, $p3$ cannot be set as true.

If $p3$ is satisfied – i.e., if both (1) and (2) are true – the following check of PU modes and the intra modes of the current depth can be skipped and mode decision process of the next depth will be conducted.

Condition $p3$ states a fact that if most of the sub residual blocks are unsmooth, current CU contains several mismatched portions. This means that the current motion vector is not sufficient to describe current CU and it should be split into smaller sized CUs.

We estimated the accuracy of conditions $p1$ and $p3$, and list them in Table 3.3. The columns indexed by p1 and p3 are the accuracy of the corresponding condition. The accuracy of condition $p1$ is measured by counting the percentage of the CUs whose best mode is decided as inter mode 2Nx2N among the CUs with residual characteristic satisfying condition $p1$. This is because that when the best mode of a CU is decided as inter mode 2Nx2N, it is considered safe to terminate the mode check process of the following candidates. The same applies to $p3$.

As shown in the table, the accuracy of condition $p1$ is as high as 97%. A single Qp value of 32 is used for this simulation. The index p2 indicates the ratio of CUs that satisfy condition $p1$ among the CUs whose best mode is decided as inter mode 2Nx2N. The index p2 can be considered as the norm used to evaluate the effectiveness of condition $p1$ from the opposite perspective. From Table 3.3, about 88% of these CUs can be terminated in advance based on condition $p1$. Index p4 in this table indicates the ratio of CUs that satisfy condition $p3$ among the CUs which are finally decided to split.

Table 3.3: Accuracy (%) of termination and split over current depth based on residual check.

Sequence		Low-delay P				Random access			
		p1	p2	p3	p4	p1	p2	p3	p4
Class A	Traffic	94.7	77.0	69.4	63.0	99.0	96.4	62.7	20.9
	PeopleOnStreet	95.6	89.9	80.2	40.9	96.3	89.6	75.1	44.7
Class B	Kimono	95.4	90.5	59.1	18.8	94.3	91.2	47.0	22.7
	ParkScene	98.6	91.2	57.0	35.9	98.6	92.8	51.9	33.5
	Cactus	98.3	91.0	50.3	46.9	98.0	92.2	47.8	46.9
	BQTerrace	98.8	83.5	29.0	58.6	99.4	84.0	15.0	53.5
	BasketballDrive	95.8	96.5	63.3	31.8	95.2	96.2	55.9	36.6
Class C	RaceHorsesC	94.8	79.4	55.7	57.6	94.2	79.8	50.2	60.1
	BQMall	98.3	88.4	54.8	48.0	98.4	90.6	52.1	45.5
	PartyScene	96.8	67.8	50.7	78.0	97.5	69.8	41.4	77.5
	BasketballDrill	97.6	92.6	66.7	38.6	97.6	93.0	62.7	38.3
Class D	RaceHorses	94.7	77.1	69.4	63.0	94.7	78.5	65.9	62.9
	BQSquare	98.4	73.2	38.8	62.0	99.3	74.0	20.5	55.2
	BlowingBubbles	97.6	79.0	50.6	62.8	97.9	81.2	44.6	58.7
	BasketballPass	98.7	93.2	68.0	41.8	98.6	93.6	63.4	42.3
Class E	Vidyo1	99.3	98.1	49.0	12.6	99.3	98.6	54.2	14.9
	Vidyo3	99.2	96.7	62.8	28.0	99.5	97.4	59.4	31.1
	Vidyo4	99.5	98.3	43.5	6.8	99.3	98.8	54.5	9.1
AVG		97.3	86.8	56.6	44.2	97.6	88.8	51.4	41.9

3.3.3 PU skip strategy

If both condition $p1$ and $p3$ are not satisfied, following PU partition modes shown in Figure 2.5 should be checked next. In order to skip some unlikely partition modes, following schemes are introduced in this chapter:

(1) If the two left or right VBs are checked as smooth, mode with Nx2N partition will be checked.
(2) If the two top or bottom VBs are checked as smooth, mode with 2NxN partition will be checked.

In the simulation, we found that this assumption is not always right. To reduce the prediction error resulted by this, the following remedy process is also proposed in this chapter. After analyzing the residual generated by mode with 2Nx2N partition, if Nx2N is decided to be checked while 2NxN is not, only mode Nx2N will be checked and its RD_Cost is compared with that of mode 2Nx2N. If cost of latter is smaller, which means the above assumption may not be true, 2NxN will still be checked. If 2NxN is decided to be checked while Nx2N is not, the same action will occur.

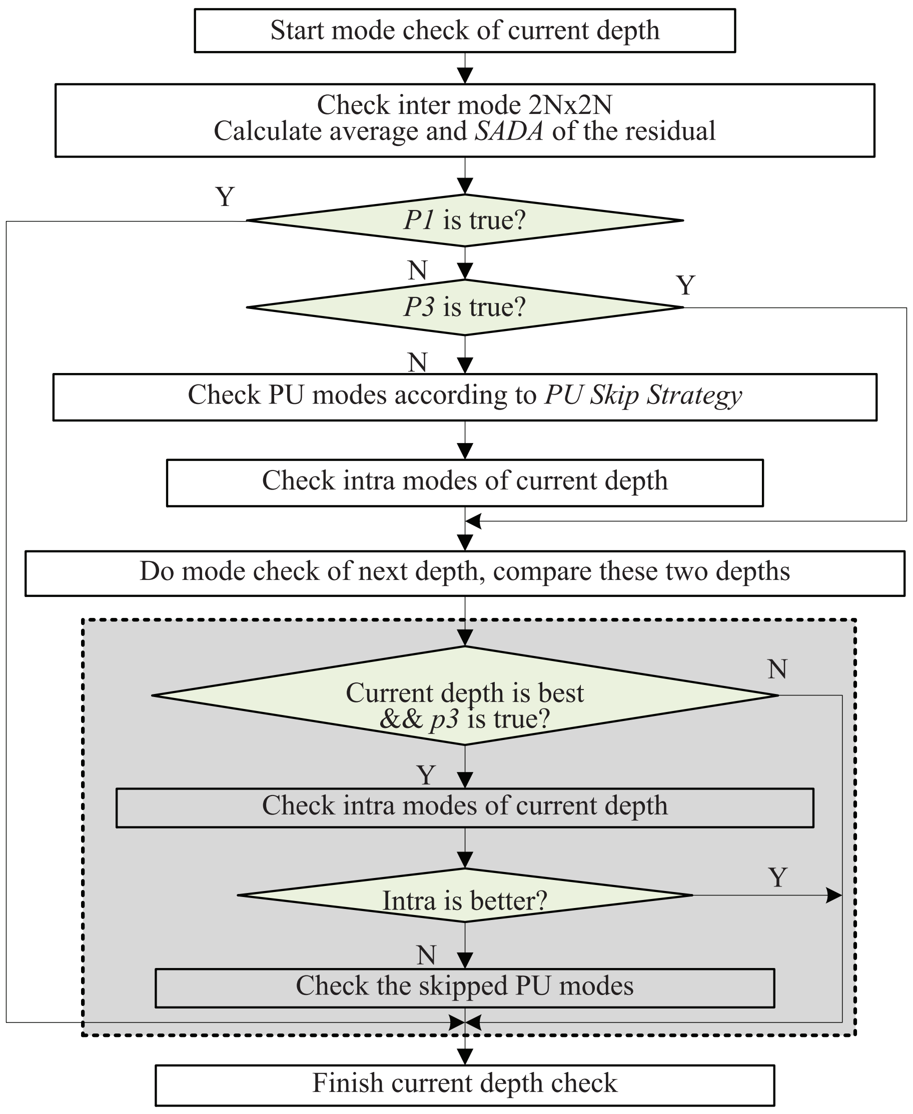
Start mode check of current depth
Check inter mode 2Nx2N
Calculate average and *SADA* of the residual
Y
P1 is true?
N
Y
P3 is true?
N
Check PU modes according to *PU Skip Strategy*
Check intra modes of current depth
Do mode check of next depth, compare these two depths
N
Current depth is best && *p3* is true?
Y
Check intra modes of current depth
Y
Intra is better?
N
Check the skipped PU modes
Finish current depth check

Figure 3.3: Implementation flowchart of the residual check algorithm.

3.3.4 Implementation of residual check algorithm

As depicted by the blocks with solid lines in Figure 1.2, for the default mode decision process in HEVC, inter mode 2Nx2N, PU modes, and intra modes are checked sequentially to find the best. Intra modes are also checked for inter frames to maintain sufficient performance, e.g., in cases where blocks with highly detailed texture are encoded. Therefore, after applying the residual check algorithm, whether or when intra prediction needs to be checked should be also considered. The integration of this algorithm into the HEVC encoder is shown in Figure 3.3.

After checking inter mode 2Nx2N, the average and SADA of the generated residual is analyzed. The current depth check will be terminated when condition $p1$ is true. If condition $p3$ is true, current CU will directly split to the next depth. Otherwise, if both $p1$ and $p3$ are false, the PU skip strategy discussed above will be adopted to skip unlikely modes, and the intra prediction modes are also checked, and then current CU will split to the next depth.

As stated in Section 1.1, when the mode check over the next depth is finished, the cost of the current depth will compete with the whole cost generated from the next depth. If $p3$ is true, and the current depth is still better, which means that one of the modes of the current depth skipped by $p3$ may be the optimal one. From Table 3.3, the accuracy of $p3$ is less than 60% on average, and the optimal mode of the current depth may be skipped when $p3$ is true. Therefore, another remedy process is also proposed. Under situation stated above, the skipped PU and intra modes will be checked again. Based on the definition of $p3$, if the next depth is not suitable, there will be a high probability that current CU contains detailed texture. Then, intra prediction modes of the current depth will be checked first. If it is better than inter mode 2Nx2N, other PU modes will still be skipped. Otherwise, all the PU modes will be checked to find the best mode with the smallest cost. The part marked by the dashed lines in Figure 3.3 shows this remedy process.

3.4 Fast intra mode decision

In practice, the intra prediction process is generally divided into two steps. First, the RMD process is conducted over 35 predefined modes to select a number of sub-optimal modes based on the Hadamard transform results and estimated mode bits. Second, the chosen subset of modes are sent to the RDO process one by one to determine the best mode with the lowest cost, denoted as RD_Cost, which is evaluated by the distortion and the encoded mode bits. Based on this two-step search strategy, the complexity can be relatively reduced without significantly decreasing the quality. For real-time application, the complexity of intra prediction is still a burden.

As is known, in most natural video sequences, the pixels are always tending to change along a certain direction, and intra prediction tool is introduced into the codec based on this idea. Therefore, many directional prediction modes are defined to exploit the spatial character of the encoding block. If the content of a block is changing along one certain direction, the difference between two neighboring pixels located within this direction will be relatively smaller than those along other directions. This direction can be called as dominant direction. Hence, if the dominant direction of a block can be detected in advance, the predefined modes whose direction is close to this direction will be chosen as the candidates for the rough mode decision process, while others will be discarded without being checked.

In previous works [27,29], two common methods were used to find the dominant direction: Sobel operator and extensive pixel based search.

3.4.1 Proposed discrete cross difference

In this work, we propose a fast DCD algorithm to detect the dominant direction. The DCD is used to estimate the difference among pixels located along a certain direction. Take a 4x4 block as an example, to reduce complexity, in this work, the DCD is computed on the basis of the cross differences between two pairs of pixels in different discrete locations. As depicted in Figure 3.4, to calculate the DCD along horizontal direction, four pixels marked by cycle are used. DCDs of four major directions of a block are calculated: horizontal (H), vertical (V), diagonal-down-right (DR), and diagonal-down-left (DL). Every four pixels marked by the same symbol are used to calculate the DCD of one certain direction.

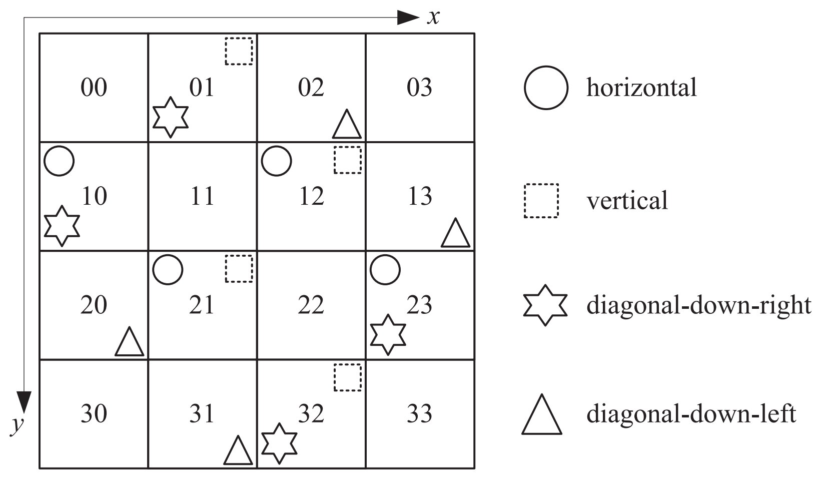

Figure 3.4: Discrete cross difference calculation pattern.

The value of DCD can be used to evaluate the strength of each direction. The strength of each direction is expressed as

$$d(H) = |f(1,2) - f(1,0)| + |f(2,3) - f(2,1)|\,, \tag{3.4}$$

$$d(V) = |f(2,1) - f(0,1)| + |f(3,2) - f(1,2)|\,, \tag{3.5}$$

$$d(DR) = |f(3,2) - f(1,0)| + |f(2,3) - f(0,1)|\,, \tag{3.6}$$

$$d(DL) = |f(2,0) - f(0,2)| + |f(3,1) - f(1,3)|\,, \tag{3.7}$$

where $f(x, y)$ represents the value of the pixel located in position (x, y).

For a block with horizontal texture, the DCD strength stated in Eq. (3.4) will be smaller than the other DCD strengths. The same holds true for the other three scenarios. Thus, the dominant direction of a block can be accurately detected by using this DCD algorithm.

In this work, the DCD strengths are calculated by the unit of a 4x4 block. For a DCD strength with certain direction of an 8x8 CU, it is calculated by making a summation of the four DCD strengths with same direction of the four 4x4 blocks. These four 4x4 blocks are located within this 8x8 CU. For even larger sized CUs (16x16), the DCD strengths can be calculated by adding up those strengths of the 8x8 CUs within it. This process is continued recursively until the size of LCU is reached. Therefore, for one LCU, the DCD calculation process is only conducted once, and the results are used by CUs at different depths.

3.4.2 Candidate mode selection

After the four DCD strengths of current CU are estimated, a subset candidate modes can be selected from the whole 35 modes based on the relationship among these four estimated directions. First, the minimum and second minimum strengths are picked out from the four DCD strengths, denoted as $miniV1$ and $miniV2$. Next, the derivation of dominant direction(s) from the DCD results can be divided into three categories:

(1) If $miniV2$ is larger than $ThV * miniV1$, it is assumed that the dominant direction of current CU is strong, and the estimated dominant direction is derived bo be the same as the direction corresponding to $miniV1$.
(2) If $miniV2$ is smaller than $ThV * miniV1$, and the directions corresponding to $miniV1$ and $miniV2$ are located next to each other – e.g., horizontal and diagonal-down-right – the dominant direction can be assumed as weak. The estimated dominant directions are derived the same as the directions corresponding to $miniV1$ and $miniV2$.
(3) If $miniV2$ is smaller than $ThV * miniV1$, and the directions corresponding to $miniV1$ and $miniV2$ are perpendicular to each other – e.g., horizontal and vertical – it is assumed that the direction of current block is not clear.

Based on the results of multiple simulations, ThV is set as 2 in order to obtain a satisfactory tradeoff between performance loss and time reduction.

The specific candidate modes according to the estimated dominant direction(s) are listed in Table 3.4. As seen in the table, the Intra_Planar and Intra_DC are always checked by the RMD process to maintain the performance. If the estimated dominant direction is derived as strong, the modes located on both sides of this direction are selected, while if it is derived as weak, the modes located between the two estimated dominant directions are chosen. Finally, if the dominant direction is not clear, only Intra_Planar and Intra_DC are checked.

Table 3.4: Candidate modes corresponding to the estimated dominate direction(s).

Category	Est. Direction(s)	Candidate modes	Total
Strong	H	0, 1, 5...15	13
	V	0, 1, 21...31	13
	DR	0, 1, 13...23	13
	DL	0, 1, 2...7, 29...34	14
Weak	H, DR	0, 1, 8..20	15
	V, DR	0, 1, 16...28	15
	H, DL	0, 1, 2...12	13
	V, DL	0, 1, 24...34	13
Not Clear	None	0, 1	2

3.4.3 Early RDO termination strategy

After the RMD process has been completed, a sub-optimal set of candidate modes are obtained and arranged in such a way that the Hadamard transformed cost increases one after another. The first mode in the set is considered as the best sub-optimal candidate. The RDO process is performed over these modes to identify the best mode with the minimum cost. RDO is typically a very time consuming process, and thus four early RDO termination strategies are proposed in this chapter:

(1) If the best sub-optimal candidate is derived as Intra_Planar or Intra_DC, only this mode will be checked by RDO, and the following candidates will be skipped.
(2) If the difference between the indexes of two neighboring candidate modes from the sub-optimal set is larger than $ThIndex$, the RDO process will be terminated before

checking the latter.

(3) If the difference between the Hadamard transformed cost of two neighboring candidate modes from the sub-optimal set is large enough, the latter and those that follow will be skipped, and the difference is estimated by deciding whether the latter is $ThRMDCost$ times larger than the former.

(4) If the RD_Cost of current candidate is $ThRDCost$ times larger than the best RD_Cost already obtained, the RDO process can be safely turned off.

Moreover, intra prediction on 4x4 blocks will be skipped when there is no significant transform coefficient after checking the 8x8 CU. The proposed fast intra mode decision algorithm consists of all the above-mentioned strategies, and the detail integration information is explained by pseudo codes as:

```
Algorithm Fast Intra Mode Decision
while current LCU != last LCU do
    for depth = 0 to 3 do
        if (depth == 0) then
            calculate the four DCD strengths for CUs in all depths;
        end if
        find the dominant direction(s);
        select the corresponding candidate modes;
        conduct the RMD process;
        for $i$th mode $\in$ sub-optimal mode set do
            if ($|ModeIndex_i - -ModeIndex_{i-1}| > ThIndex$) then
                break;
            end if
            if ($RMDCost_i > ThRMDCost * RMDCost_{i-1}$) then
                break;
            end if
            conduct the RDO process of this mode;
            if ($RDCost_i$ >best $RDCost$ already obtained) then
                break;
            end if
            if (($ModeIndex_i == 0$) || ($ModeIndex_i == 1$)) then
                break;
            end if
        end for
        if (depth == 3) then
            check intra prediction on 4x4 blocks, if needed;
        end if
    end for
end while
```

3.5 Overall algorithm

In this section, the three aforementioned algorithms are integrated to form the proposed hierarchical structure based fast mode decision algorithm. These three sub-algorithms are: the depth prediction algorithm, used to decide the size of a block based on depth information of co-located LCU; the residual check algorithm, designed to skip some unnecessary modes on the basis of the residual character; the fast intra mode decision algorithm, used to reduce the number of modes checked by RMD process according to the dominant direction. The detail integration information is given in the flowchart depicted in Figure 3.5. The overall algorithm is explained stepwise as following:

Step 1. Start a new LCU encoding process.
Step 2. Read the saved depth information of the co-located LCU in the previous frame, if it is available.
Step 3. Start the mode decision process at the current depth. If current depth is equal to 0, go to step 4, otherwise, go to step 6.
Step 4. Calculate the DCD strength of the predefined four directions for all CUs at all depths. Go to step 5.
Step 5. If the co-located LCU of current LCU is complex, go to step 20. Otherwise, go to step 8.
Step 6. If the current depth is equal to 3, go to step 7. Otherwise, go to step 8.
Step 7. If the co-located LCU of current LCU is simple, go to step 20, otherwise, go to step 8.
Step 8. Check the inter mode 2Nx2N of current CU and calculate the average and SADA of the generated residual. Go to step 9.
Step 9. If the termination condition $p1$ (Section 3.3.2) is true, go to step 20. Otherwise, go to step 10.
Step 10. If the split condition $p3$ defined in Section 3.3.2 is true, go to step 15. Otherwise, go to step 11.
Step 11. Skip unnecessary PU modes based on the PU skip strategy (Section 3.3.3). Go to step 12.
Step 12. Find the corresponding dominant direction(s) based on the DCD strengths and select a set of candidate modes. Go to step 13.
Step 13. Conduct the RMD process for this set of candidate modes to find the sub-optimal modes. Go to step 14.
Step 14. Conduct the RDO process over this set of sub-optimal modes combined with the early termination conditions defined in Section 3.4.3. Go to step 15.
Step 15. Call the recursive CU splitting process and perform mode checks at the next depth, then compare the costs of these two depths. Go to step 16.
Step 16. If the cost of current depth is smaller, and condition p3 is true, go to step 17. Otherwise, go to step 20.
Step 17. Check the intra prediction modes of current CU. Go to step 18.
Step 18. If cost of the best intra mode is smaller than that of inter mode 2Nx2N, go to step 20. Otherwise, go to step 19.
Step 19. Check the skipped PU modes sequentially. Go to step 20.
Step 20. Finish the encoding process over the current depth. Go to step 21.
Step 21. If the current depth is equal to 3, go to step 22. Otherwise, return to step 3.
Step 22. Finish the encoding process of current LCU, and store the maximum depth information.

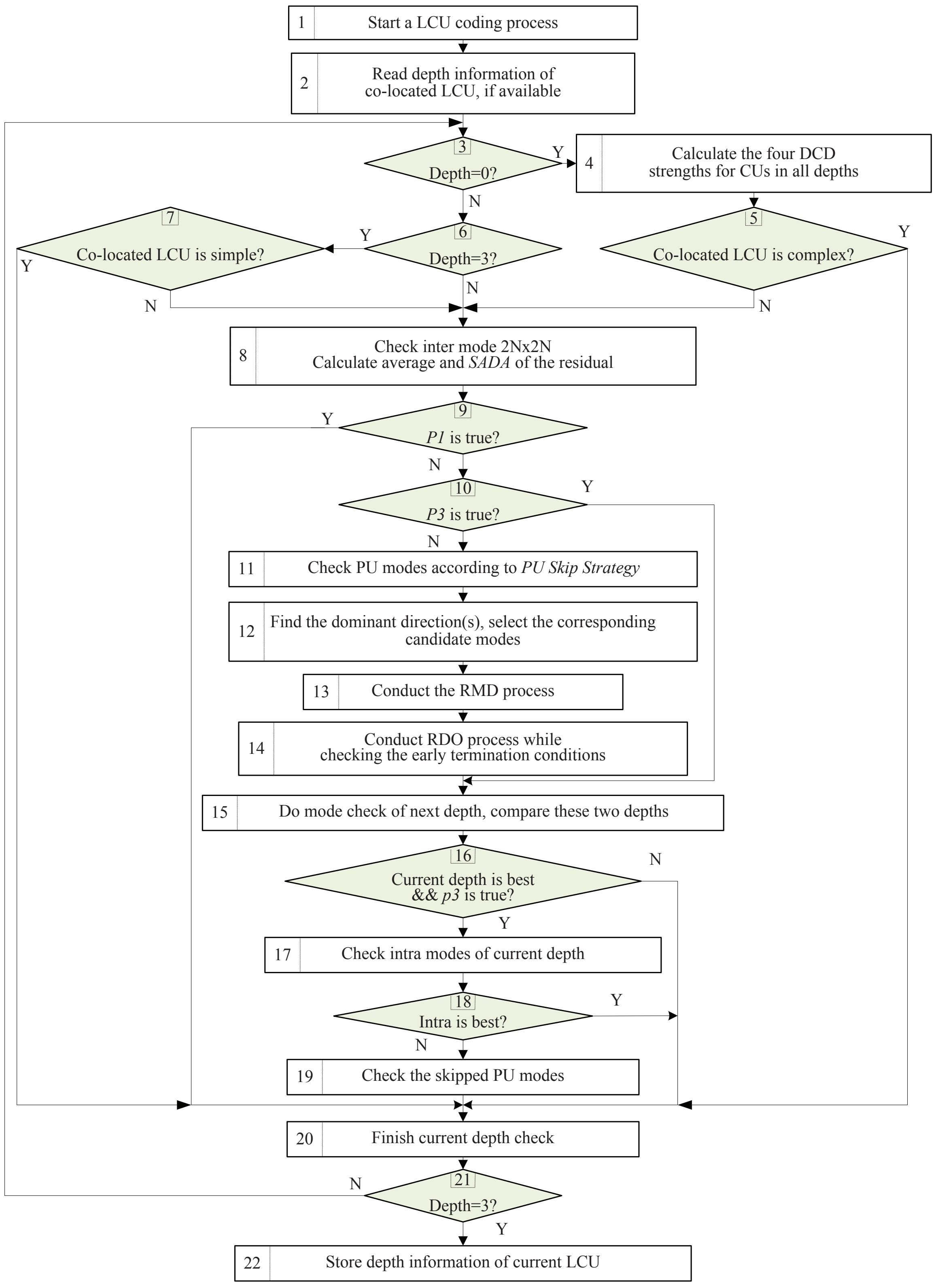
1 Start a LCU coding process
2 Read depth information of co-located LCU, if available
3 Depth=0?
Y
4 Calculate the four DCD strengths for CUs in all depths
N
5 Co-located LCU is complex?
6 Depth=3?
7 Co-located LCU is simple?
Y
N
8 Check inter mode 2Nx2N Calculate average and SADA of the residual
9 P1 is true?
10 P3 is true?
11 Check PU modes according to PU Skip Strategy
12 Find the dominant direction(s), select the corresponding candidate modes
13 Conduct the RMD process
14 Conduct RDO process while checking the early termination conditions
15 Do mode check of next depth, compare these two depths
16 Current depth is best && p3 is true?
17 Check intra modes of current depth
18 Intra is best?
19 Check the skipped PU modes
20 Finish current depth check
21 Depth=3?
22 Store depth information of current LCU

Figure 3.5: Flowchart of the proposed overall algorithm.

3.6 Simulation results

The proposed algorithms are implemented into the HEVC reference software HM11.0 and simulation specifications are provided in [123]. Three common test conditions, namely All intra, Low-delay P, and Random access, are used for evaluations.

The comparison results are evaluated based on the difference of coding time (ΔT - indicating the average time reduction in the coding process), the PSNR difference ($\Delta PSNR$ - denoting the average difference of the peak signal-to-noise ratio), and the bitrate difference (ΔBR - indicating the average bit-rate increase). $\Delta PSNR$ and ΔBR are calculated according to [124]. The test platform used is Intel Core i7-3.4GHz, 8.0GB RAM. The above-mentioned parameters are defined as

$$\Delta T = \frac{T_{proposed} - T_{reference}}{T_{reference}} \times 100\%, \tag{3.8}$$

$$\Delta PSNR = PSNR_{proposed} - PSNR_{reference}, \tag{3.9}$$

$$\Delta BR = \frac{BR_{proposed} - BR_{reference}}{BR_{reference}} \times 100\%. \tag{3.10}$$

3.6.1 Performance of depth prediction

As discussed in section 3.2.3, depth prediction is turned off once for every N frames. In this work, N is decided as 16 according to some simulation results. If a number larger than 16 is chosen, the performance of depth prediction will slightly decrease, while time saving by this algorithm will increase due to the fact that more frames can be predicted by using depth prediction. On the other hand, the opposite is true for a number smaller than 16.

The evaluation results of the proposed depth prediction algorithm are tabulated in Table 3.5. In this table, the bitrate and PSNR loss as well as the encoding time reduction of the proposed algorithm are given. In terms of complexity reduction, this proposed algorithm can save 41% encoding time under All Intra condition, 42.9% under Low-delay P, and 42.6% under Random access, on average. The largest complexity reduction is 57% when encoding sequence Kimono under All Intra condition. On the other hand, the average bitrate increases are 0.001% under All Intra, 0.084% under Low-delay P, and 0.342% under Random access conditions.

Consider the encoding time reduction, the average savings are almost same under all three test conditions. This is because the sum ratios of simple and complex LCUs always hold a high percentage under all these conditions, as shown in Table 3.1.

With regards to the bitrate, the bitrate increase under All intra condition is smaller than the other two. For the Low-delay P case, the bitrate increase may come from the fact that different frames are used for motion estimation and depth prediction process. As mentioned in section 3.2.2, in order to reduce the complexity for saving the depth information, only the frame right before current frame in encoding order is used by the depth prediction. With the Random access case, the above mentioned difference is even larger. For under this condition, reference frames with a larger temporal distance, compared with Low-delay P, are also used. Therefore, the bitrate loss is the largest among the three conditions.

Table 3.5: Simulation results for the proposed depth prediction algorithm.

Sequence		All intra					Low-delay P					Random access				
		ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT
			Y	U	V			Y	U	V			Y	U	V	
Class A	Traffic	0.008	-0.001	0.000	0.000	-40.4	0.498	-0.008	0.012	0.018	-46.1	0.367	-0.019	-0.001	-0.004	-47.1
	PeopleOnStreet	-0.010	-0.001	-0.002	0.004	-38.6	0.065	-0.004	0.003	-0.004	-39.4	0.124	-0.008	0.007	-0.003	-35.8
Class B	Kimono	-0.024	-0.002	0.005	-0.001	-57.0	-0.092	0.000	-0.007	-0.005	-42.2	-0.030	-0.006	-0.006	0.001	-44.3
	ParkScene	-0.033	-0.002	-0.002	-0.002	-40.9	0.128	-0.014	0.009	0.009	-44.0	0.133	-0.018	-0.005	-0.007	-44.7
	Cactus	-0.021	-0.002	0.000	-0.002	-40.0	0.080	-0.007	0.005	0.003	-44.0	0.686	-0.012	0.002	-0.003	-45.0
	BQTerrace	-0.003	0.000	0.001	0.005	-43.7	-0.385	-0.010	0.008	0.006	-44.9	-0.034	-0.007	0.004	0.002	-46.6
	BasketballDrive	0.101	-0.001	-0.006	0.001	-46.3	0.162	-0.001	-0.005	-0.008	-42.7	0.488	-0.011	-0.026	-0.012	-43.6
Class C	RaceHorsesC	-0.027	-0.002	-0.002	-0.001	-37.3	-0.025	-0.006	-0.002	-0.002	-40.4	0.456	-0.015	-0.016	-0.030	-38.9
	BQMall	-0.013	0.000	0.002	-0.006	-39.1	-0.108	-0.006	-0.016	-0.030	-42.0	0.377	-0.021	-0.006	0.029	-40.9
	PartyScene	-0.001	0.000	0.001	0.001	-36.8	0.054	-0.009	0.028	-0.005	-39.4	0.253	-0.013	0.005	-0.001	-41.1
	BasketballDrill	-0.012	-0.001	0.004	-0.003	-38.2	-0.089	-0.011	-0.012	0.025	-43.2	0.318	-0.012	-0.023	-0.027	-42.4
Class D	RaceHorses	-0.008	-0.001	-0.001	0.001	-36.2	0.283	0.006	-0.017	-0.038	-36.0	0.413	-0.013	0.021	0.005	-27.2
	BQSquare	0.000	0.000	0.000	0.000	-35.9	0.308	-0.011	0.001	-0.031	-37.7	0.583	-0.005	0.010	0.006	-42.6
	BlowingBubbles	-0.011	0.000	0.003	0.002	-34.5	0.142	-0.036	0.041	-0.023	-37.6	0.430	-0.013	-0.002	-0.003	-40.5
	BasketballPass	0.029	-0.001	0.012	0.002	-38.5	0.320	-0.012	0.017	0.035	-40.7	0.344	-0.023	-0.022	-0.029	-40.3
Class E	Vidyo1	0.005	0.001	0.004	0.002	-42.5	-0.136	-0.014	0.006	0.031	-51.3	0.505	-0.005	-0.002	0.008	-49.1
	Vidyo3	0.045	-0.005	-0.004	0.015	-47.2	0.266	-0.014	-0.034	-0.014	-50.6	0.342	-0.014	0.008	0.007	-49.4
	Vidyo4	-0.006	-0.002	0.009	0.010	-44.8	0.040	-0.004	-0.020	-0.013	-50.4	0.399	-0.010	-0.001	-0.004	-46.7
AVG		0.001	-0.001	0.001	0.002	-41.0	0.084	-0.009	0.001	-0.002	-42.9	0.342	-0.013	-0.003	-0.004	-42.6

RD curves of PartyScene and BlowingBubbles are depicted in Figure 3.6 and 3.7 as examples. The sequences are encoded by the proposed depth prediction algorithm under Low-delay P test condition, calculated with Qp set as 28, 32, 36, and 40. These curves have shown that the proposed method has the similar RDO performance as that of HM.

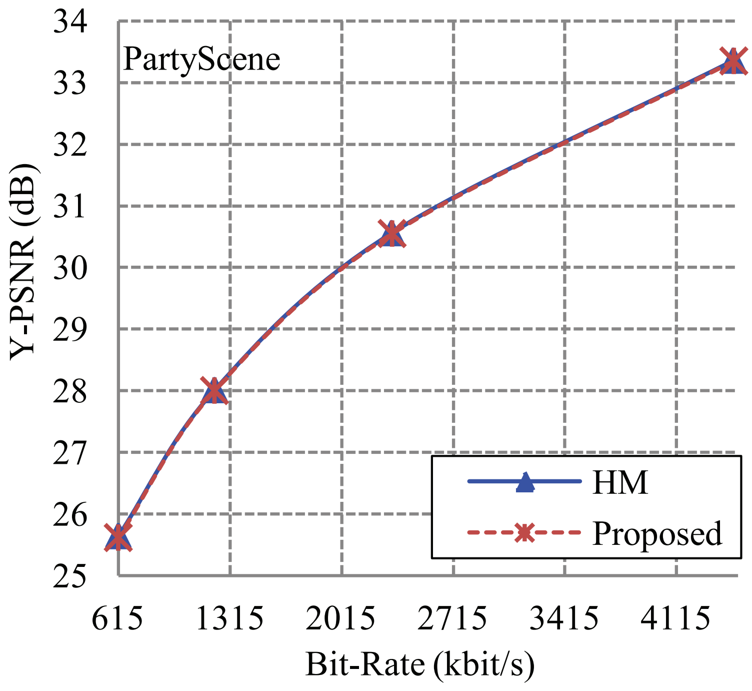

Figure 3.6: RD curve of PartyScene.

3.6.2 Performance of depth prediction combined with residual check

To decide the values of the thresholds, $ThSmooth$ and $ThAvg$, some simulations are conducted. A series of combinations of d and $ThAvg$ are tested for the proposed residual check algorithm with termination condition $p1$. Some sequences from Class C and Class D are used. These simulations are conducted under low-delay P with a single Qp value of 32. The results are tabulated in Table 3.6.

Table 3.6: Impact of the selection of $ThSmooth$ and $ThAvg$.

d	$ThAvg$	Class C			Class D		
		ΔBR	$\Delta PSNR$	ΔT	ΔBR	$\Delta PSNR$	ΔT
1	4	-0.09	-0.01	21.2	-0.08	-0.01	20.7
3	6	-0.07	-0.02	31.3	-0.00	-0.01	28.9
4	7	-0.03	-0.02	37.5	0.03	-0.02	34.0
5	8	0.01	-0.05	43.3	0.13	-0.03	39.0
7	10	0.18	-0.12	51.4	0.34	-0.15	48.7

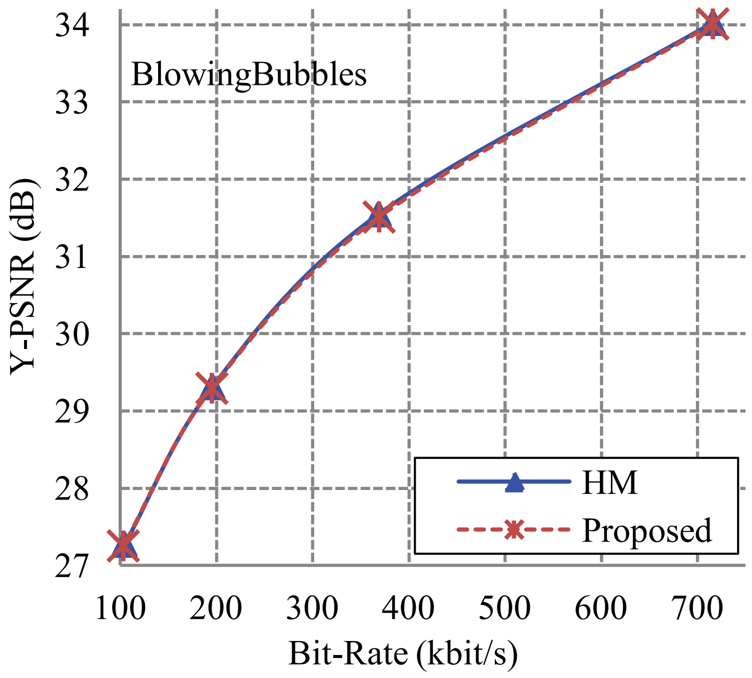

Figure 3.7: RD curve of BlowingBubbles.

According to Eq. (3.3), the value of $ThSmooth$ becomes larger when d increases. From the table, we can see that along with the increasing of $ThSmooth$ and $ThAvg$, the bitrate loss as well as the PSNR drop will increase at the same time, but, the corresponding time saving will be more substantial. On the other hand, for certain fixed thresholds, the impact on different Classes is almost the same in terms of the performance of time reduction. Therefore, to maintain a tradeoff between performance and complexity reduction d and $ThAvg$ are decided as 4, and 7, respectively.

The evaluation results of the proposed residual check algorithm combined with the depth prediction algorithm are tabulated in Table 3.7. For the residual check algorithm is proposed based on inter frame prediction case, so there is no need to be evaluated under All Intra test condition. In this table, the bitrate and PSNR loss as well as the encoding time reduction of the combined algorithm are given.

In terms of complexity reduction, this combined algorithm can save 69.4% encoding time under Low-delay P test condition, and 72.4% under Random access, on average. The largest complexity reduction is 85.3% when encoding sequence Vidyo1 under Low-delay P condition. On the other hand, the average bitrate increases are 1.87% under Low-delay P, and 1.70% under Random access conditions. The bitrate loss is marginal compared with the reduction in encoding time.

In terms of time reduction, the average savings are almost the same for both test conditions. On the part of bitrate loss, the bitrate increase when encoding the sequences Kimono and Cactus listed in Table VI, is relatively larger than the other test sequences. This is because the contents of these two sequences include highly detailed textures. Hence the generated residual tends to be irregular, and its character is not easy to be predicted by the residual check conditions.

Table 3.7: Simulation results for the proposed depth prediction combined with residual check algorithm.

Sequence		Low-delay P					Random access				
		ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT
			Y	U	V			Y	U	V	
Class A	Traffic	0.359	-0.146	-0.051	-0.051	-77.8	0.346	-0.111	-0.014	-0.007	-81.1
	PeopleOnStreet	1.021	-0.086	-0.037	-0.019	-65.1	1.199	-0.072	-0.010	0.006	-66.5
Class B	Kimono	3.322	-0.047	-0.041	-0.026	-71.0	3.643	-0.052	-0.023	-0.005	-73.2
	ParkScene	0.298	-0.082	-0.021	-0.006	-71.9	-0.309	-0.092	-0.013	-0.011	-76.1
	Cactus	4.443	-0.068	-0.030	-0.045	-71.9	2.505	-0.057	-0.003	-0.014	-75.1
	BQTerrace	2.826	-0.073	-0.019	-0.019	-71.1	1.694	-0.069	0.013	0.014	-71.5
	BasketballDrive	2.442	-0.051	-0.033	-0.076	-73.2	2.573	-0.049	-0.036	-0.059	-75.7
Class C	RaceHorsesC	1.416	-0.036	-0.065	-0.102	-65.7	1.740	-0.067	-0.067	-0.084	-66.1
	BQMall	3.023	-0.081	-0.047	-0.052	-67.7	1.983	-0.107	-0.022	-0.039	-71.0
	PartyScene	1.954	-0.065	-0.013	-0.044	-57.1	1.678	-0.107	-0.045	-0.051	-62.2
	BasketballDrill	1.782	-0.081	-0.057	-0.082	-66.8	1.951	-0.069	-0.049	-0.081	-73.3
Class D	RaceHorses	1.416	-0.044	-0.059	-0.086	-58.8	1.705	-0.051	-0.033	-0.056	-62.5
	BQSquare	0.923	-0.045	-0.016	-0.004	-57.8	2.952	-0.130	-0.006	-0.002	-62.9
	BlowingBubbles	1.464	-0.044	-0.026	-0.008	-54.7	1.201	-0.074	-0.046	-0.046	-62.2
	BasketballPass	0.850	-0.068	-0.081	-0.050	-70.7	1.146	-0.072	-0.003	-0.047	-71.9
Class E	Vidyo1	1.335	-0.224	-0.047	-0.056	-84.5	1.293	-0.122	0.067	0.071	-85.3
	Vidyo3	2.236	-0.156	-0.017	-0.032	-81.1	2.054	-0.085	0.042	0.079	-83.4
	Vidyo4	2.582	-0.130	-0.032	-0.012	-83.0	1.230	-0.087	0.099	0.085	-82.9
AVG		1.872	-0.085	-0.038	-0.043	-69.4	1.699	-0.082	-0.008	-0.014	-72.4

3.6.3 Performance of fast intra mode decision

Based on the results of multiple simulations, ThV is set as 2 in order to obtain a satisfactory tradeoff between performance loss and time reduction. If ThV is set larger than 2, the performance of this algorithm will slightly decrease, while time saving will increase. For the number of blocks with dominant direction decided as not clear will increase under such kind of threshold. On the other hand, the opposite is true for the ThV smaller than 2. Other thresholds, $ThIndex$, $ThRMDCost$, and $ThRDCost$, are selected as 3, 1.5, and 1.2, respectively. According to simulation results, smaller values of these three thresholds may decrease the performance with more time reductions. For a fewer number of modes will be checked under thus condition. On the other hand, the opposite is true for larger thresholds.

Table 3.8 shows the simulation results of the proposed fast intra mode decision algorithm. In this table, the bitrate and PSNR loss as well as the encoding time reduction of the proposed algorithm are tabulated.

In terms of complexity reduction, this proposed algorithm can save 53.2% encoding time under All Intra, 25.1% under Low-delay P, and 24.0% under Random access conditions, on average. The largest complexity reduction is 60.0% when encoding sequence Vidyo3 under All Intra condition. This is because the inter mode decision process is much more complex than that of intra mode decision process due to the motion estimation in inter prediction. Hence, the time to perform inter mode decision is much longer than that of intra prediction, and time saving obtained by the proposed fast intra mode decision algorithm is relatively small in Low-delay P and Random access conditions.

On the other hand, the average bitrate increases are 2.44% under All Intra, 0.51% under Low-delay P, and 0.15% under Random access conditions, on average. It should be noted that, CUs in inter predicted fames, can choose both the intra and inter modes as possible candidates, and this proposed algorithm only affects the performance of intra modes, so the bitrate losses in both Low-delay P and Random access cases are relatively small.

3.6.4 Performance of overall algorithm

In Table 3.9, the performance comparison of the proposed overall algorithm with HM is tabulated. In this table, the bitrate and PSNR loss along with the complexity reduction of the proposed algorithms are listed.

With regards to the encoding time reduction, the overall algorithm achieves a similar performance as the combined algorithm of depth prediction and residual check. As shown in Figure 3.5, when encoding a new LCU, the DCD strengths of the four predefined directions are firstly calculated for all the CUs with different depths. However, after the depth prediction and residual check algorithms being applied, the intra prediction process of most CUs is skipped, so the time saving obtained from the proposed fast intra mode decision algorithm is very small, meanwhile the time consumed by DCD strength calculation offsets the time saving to an even more smaller percentage. Therefore, the contribution of the fast intra prediction algorithm tends to be smaller than the other two algorithms under Low-delay P and Random access conditions.

Therefore, we propose to for the All intra case, use the depth prediction algorithm combined with the fast intra prediction algorithm, while for the Low-delay P and Random access test conditions, use the combined algorithm of depth prediction and residual check algo-

Table 3.8: Simulation results for the proposed fast intra mode decision algorithm.

Sequence		All intra					Low-delay P					Random access				
		ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT
			Y	U	V			Y	U	V			Y	U	V	
Class A	Traffic	2.236	-0.085	0.003	0.005	-52.2	-	-	-	-	-	0.129	-0.013	0.004	0.000	-26.5
	PeopleOnStreet	3.174	-0.084	0.002	0.005	-52.6	-	-	-	-	-	0.372	-0.006	-0.004	0.008	-21.6
Class B	Kimono	1.123	-0.041	-0.009	-0.014	-53.2	0.179	-0.003	-0.001	-0.005	-31.5	0.091	-0.001	-0.001	0.001	-25.4
	ParkScene	0.150	-0.075	0.010	0.001	-52.4	0.062	-0.008	-0.002	0.003	-31.2	-0.223	-0.008	0.013	0.002	-25.5
	Cactus	2.272	-0.066	0.002	-0.012	-53.0	0.490	-0.009	0.002	0.002	-31.3	0.344	-0.008	-0.001	0.002	-25.2
	BQTerrace	1.788	-0.076	0.011	0.010	-55.3	0.331	-0.012	0.002	-0.001	-29.9	0.042	-0.008	-0.005	0.008	-22.7
	BasketballDrive	3.408	-0.044	0.007	0.005	-55.8	1.336	-0.007	-0.003	-0.025	-21.5	0.686	-0.008	-0.018	-0.009	-23.3
Class C	RaceHorsesC	1.527	-0.094	-0.009	-0.013	-51.0	0.849	-0.005	-0.026	-0.031	-28.6	0.233	-0.015	-0.008	-0.004	-25.6
	BQMall	2.421	-0.120	0.001	0.001	-52.0	0.373	-0.023	-0.024	-0.006	-27.4	0.002	-0.011	-0.015	-0.024	-23.5
	PartyScene	1.206	-0.156	-0.006	-0.007	-48.8	0.448	-0.021	0.001	-0.005	-20.2	0.218	-0.016	0.009	0.003	-22.9
	BasketballDrill	2.852	-0.085	-0.004	-0.008	-53.7	0.558	-0.028	0.002	-0.019	-19.8	-0.085	-0.024	0.009	-0.011	-24.7
Class D	RaceHorses	2.084	-0.109	-0.011	-0.010	-49.8	0.777	-0.020	-0.025	-0.035	-21.5	0.217	-0.006	-0.005	0.006	-23.8
	BQSquare	1.883	-0.155	-0.014	-0.015	-52.3	0.140	-0.022	-0.039	0.004	-21.3	0.029	-0.020	0.037	0.013	-23.7
	BlowingBubbles	1.743	-0.118	-0.015	-0.011	-48.0	0.138	-0.014	-0.023	0.006	-16.1	-0.033	-0.013	-0.020	-0.008	-22.6
	BasketballPass	3.544	-0.094	-0.016	-0.009	-51.7	0.431	-0.029	-0.048	0.006	-29.9	0.200	-0.031	0.004	0.022	-22.9
Class E	Vidyo1	4.905	-0.106	0.016	0.014	-57.7	0.698	-0.018	0.028	0.014	-24.1	-	-	-	-	-
	Vidyo3	3.794	-0.084	0.004	0.027	-60.0	0.631	-0.012	-0.028	-0.046	-19.3	-	-	-	-	-
	Vidyo4	3.893	-0.089	0.009	-0.009	-57.9	0.779	-0.026	-0.059	-0.027	-28.7	-	-	-	-	-
AVG		2.445	-0.093	-0.001	-0.002	-53.2	0.514	-0.016	-0.015	-0.010	-25.1	0.148	-0.012	0.000	0.001	-24.0

Table 3.9: Simulation results for the proposed overall algorithm.

Sequence		All intra					Low-delay P					Random access				
		ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT
			Y	U	V			Y	U	V			Y	U	V	
Class A	Traffic	2.223	-0.085	0.003	0.004	-53.3	-	-	-	-	-	0.410	-0.111	-0.011	-0.007	-79.9
	PeopleOnStreet	3.201	-0.081	0.001	0.000	-51.6	-	-	-	-	-	1.558	-0.072	-0.002	0.009	-64.8
Class B	Kimono	1.161	-0.041	-0.006	-0.011	-66.6	3.487	-0.054	-0.036	-0.020	-69.7	3.833	-0.056	-0.030	-0.009	-71.9
	ParkScene	0.190	-0.073	0.010	0.003	-53.6	0.473	-0.082	-0.023	-0.002	-70.3	-0.334	-0.090	-0.012	-0.009	-74.7
	Cactus	2.291	-0.064	0.000	-0.011	-53.8	4.928	-0.066	-0.034	-0.049	-70.4	2.722	-0.058	-0.001	-0.015	-73.8
	BQTerrace	1.807	-0.074	0.013	0.011	-55.8	2.779	-0.069	-0.021	-0.015	-69.3	1.670	-0.070	0.011	0.010	-69.7
	BasketballDrive	3.423	-0.046	-0.001	0.000	-59.1	3.648	-0.053	-0.047	-0.101	-71.9	3.269	-0.049	-0.031	-0.061	-74.5
Class C	RaceHorsesC	1.545	-0.091	-0.008	-0.009	-51.1	2.114	-0.035	-0.084	-0.110	-64.3	1.827	-0.066	-0.073	-0.100	-64.2
	BQMall	2.391	-0.117	0.002	-0.006	-52.8	4.025	-0.091	-0.048	-0.059	-66.1	2.175	-0.103	-0.021	-0.042	-69.4
	PartyScene	1.182	-0.151	-0.005	-0.006	-49.6	2.352	-0.059	-0.020	-0.048	-55.3	1.961	-0.108	-0.048	-0.052	-60.1
	BasketballDrill	3.170	-0.078	-0.001	-0.007	-51.1	2.094	-0.091	-0.066	-0.112	-65.2	2.154	-0.067	-0.055	-0.071	-71.9
Class D	RaceHorses	2.042	-0.106	-0.022	-0.019	-49.3	1.990	-0.041	-0.084	-0.094	-57.0	1.743	-0.057	-0.062	-0.060	-60.3
	BQSquare	1.909	-0.147	-0.003	-0.014	-49.8	1.124	-0.050	-0.007	-0.036	-55.3	2.940	-0.128	0.006	0.000	-60.5
	BlowingBubbles	1.744	-0.116	-0.002	-0.015	-47.6	1.648	-0.042	-0.022	0.007	-52.4	1.508	-0.070	-0.041	-0.040	-60.2
	BasketballPass	3.550	-0.099	-0.013	0.001	-52.0	1.254	-0.072	-0.091	-0.054	-69.2	1.392	-0.068	-0.011	-0.062	-70.3
Class E	Vidyo1	5.078	-0.103	0.013	0.012	-57.5	1.426	-0.218	-0.062	-0.049	-83.5	-	-	-	-	-
	Vidyo3	4.302	-0.090	0.001	0.039	-59.7	2.298	-0.158	-0.006	-0.009	-80.0	-	-	-	-	-
	Vidyo4	3.970	-0.088	0.004	-0.002	-58.2	2.462	-0.135	-0.035	-0.029	-82.0	-	-	-	-	-
AVG		2.510	-0.092	-0.001	-0.002	-54.0	2.381	-0.082	-0.043	-0.049	-67.6	1.922	-0.078	-0.025	-0.034	-68.4

rithms.

3.6.5 Performance compared with previous works

The performance of the proposed depth prediction algorithm is compared with that of several previous works, listed in [39–42, 44, 45]. The complexity reduction and corresponding bitrate loss are adopted as criteria to make a reasonable comparison. The performance comparison results are tabulated in Table 3.10. Results show that the proposed algorithm achieves better effect in terms of encoding time reduction with a negligible bitrate increase.

In reference [41], co-located LCUs from reference frames are used, and depth information of all possible reference frames has to be stored. While our proposed depth prediction only uses the depth information of the co-located LCU from the previous frame in an encoding order. Therefore, reference [41] can serve as a sensible benchmark. According to the results, this proposed algorithm achieves more time savings with similar bitrate loss. Compared with other works, this algorithm also achieves more time savings with less bitrate loss.

Then, the performance of the proposed residual check (3.3) is also analyzed. In previous work, such as reference [58], the residual information is only used to terminate the mode decision process and such well-established concept can serve as a benchmark. Therefore, we implement the residual check only with CU termination (condition $p1$) as a sensible benchmark to evaluate other strategies such as split condition $p3$, PU skip strategy, and remedy process. The results are given in Table 3.11. Results show that this proposed algorithm can save about 40% encoding time with negligible bitrate loss compared with the above mentioned benchmark.

3.7 Conclusion

In this chapter, a course of hierarchical structure based low complexity fast mode decision algorithms were presented. Firstly, the co-located depth information from previous frame was used to predict the split structure of current LCU. Then, the residual generated by inter prediction was analyzed to determine the encoding flow: terminating the encoding process or directly skipping to the next depth of the quadtree, or eliminating some unnecessary modes. Finally, a hardware-oriented low complexity fast intra prediction algorithm was proposed. This proposed algorithm adopted a fast DCD to detect the dominant direction of the CU. Moreover, four simple but efficient early termination strategies were proposed to terminate the RDO process properly. Simulation results showed that the proposed overall algorithm reduced the encoding time by $54.0 \sim 68.4\%$, without introducing any noticeable performance degradation.

Table 3.10: Performance of the depth prediction compared with previous works.

Condition	Method	Class A	Class B	Class C	Class D	Class E	AVG
		(ΔBR,ΔT)	(ΔBR,ΔT)	(ΔBR,ΔT)	(ΔBR,ΔT)	(ΔBR,ΔT)	
All intra	Proposed	(0.00, -39.5)	(0.00, -45.6)	(-0.01, -37.8)	(0.00, -36.3)	(0.01, -44.8)	(0.00, -40.8)
	Ref. [39]	(0.20, -25.4)	(0.20, -21.0)	(0.10, -14.5)	(0.00, -16.1)	(0.20, -26.0)	(0.14, -20.6)
	Ref. [40]	(2.28, -22.0)	(2.16, -28.6)	(1.50, -17.8)	(1.20, 14.5)	(-, -)	(1.79, -13.5)
	Ref. [41]]	(0.00, -14.7)	(0.00, -19.4)	(0.00, -12.0)	(0.00, -14.7)	(0.00, -19.3)	(0.00, -16.0)
Low-delay P	Proposed	(0.28, -42.8)	(-0.02, -43.5)	(-0.04, -41.2)	(0.26, -38.0)	(0.06, -50.7)	(0.11, -43.2)
	Ref. [41]	(-, -)	(0.40, -30.9)	(0.40, -24.7)	(0.30, -16.9)	(0.50, -38.7)	(0.40, -27.8)
	Ref. [42]	(-, -)	(0.80, -34.7)	(1.20, -28.4)	(1.06, -16.3)	(1.06, -41.3)	(1.03, -30.2)
	Ref. [44]	(-, -)	(0.57, -27.5)	(0.43, -21.0)	(0.25, -18.8)	(0.83, -43.3)	(0.52, -27.7)
Random access	Proposed	(0.25, -41.5)	(0.25, -44.8)	(0.35, -40.8)	(0.44, -37.6)	(0.42, -48.4)	(0.34, -42.6)
	Ref. [41]	(0.70, -22.0)	(0.80, -25.5)	(0.50, -19.8)	(0.30, -16.5)	(-, -)	(0.58, -21.0)
	Ref. [42]	(1.34, -32.5)	(1.02, -36.3)	(1.41, -29.8)	(0.99, -18.0)	(-, -)	(1.19, -29.2)
	Ref. [44]	(-, -)	(0.08, -16.5)	(0.11, -13.0)	(0.14, -12.5)	(0.07, -31.3)	(0.10, -18.3)
	Ref. [45]	(-, -)	(0.89, -6.0)	(0.75, -12.1)	(0.59, -12.6)	(-, -)	(0.74, -10.2)

Table 3.11: Performance of the residual check.

Sequence	Low-delay P					Random access				
	ΔBR	$\Delta PSNR$			ΔT	ΔBR	$\Delta PSNR$			ΔT
		Y	U	V			Y	U	V	
Class A	0.612	-0.001	-0.005	-0.006	-41.8	0.553	-0.005	-0.005	0.002	-40.6
Class B	2.353	-0.006	-0.001	-0.004	-39.3	2.107	-0.007	-0.012	-0.012	-37.7
Class C	2.192	0.000	-0.013	-0.012	-45.0	1.677	-0.012	-0.017	-0.018	-44.0
Class D	1.135	0.003	-0.014	-0.006	-45.8	1.311	-0.005	-0.020	-0.022	-43.0
Class E	0.244	-0.002	0.000	-0.005	-35.4	0.223	-0.009	-0.001	-0.001	-33.7
AVG	1.307	-0.001	-0.007	-0.007	-41.5	1.174	-0.008	-0.011	-0.010	-39.8

Chapter 4

Hardware architecture of the fast mode decision algorithm for H.265/HEVC

4.1 Introduction

In this chapter, the corresponding hardware architectures of the proposed fast mode decision algorithms in Chapter 3 are presented. First, a state machine based module for the depth prediction combined with the residual check algorithm is described [119]. The system framework where this proposed module located is introduced and the realization of the internal architecture of this module is analyzed. Furthermore, for the proposed state machine, there are 13 states are defined and the transition conditions are discussed. Then, the hardware implementation of the proposed fast DCD algorithm and two previous works is given [120]. Finally, these hardware architectures are synthesised and the results are also provided. Moreover, the complexity and performance of the proposed DCD algorithm are compared with previous works.

4.2 Hardware architecture of the depth prediction combined with residual check

4.2.1 Proposed mode dispatch system

In order to achieve a better compatibility, in this work, we propose to design the fast mode decision algorithms as an individual module that can be easily embedded into a common video codec for H.265/HEVC. Hence, the input and output interfaces between the proposed hardware architecture and the video codec system are designed based on simple handshake protocol. The framework of the video codec system where our proposed module located is illustrated in Figure 4.1.

The framework of this system mainly contains 5 parts: the picture buffer, the inter/intra prediction coding engine, the transform and entropy coding engine, the mode cost calculation engine, and the state machine based fast mode decision module along with its surrounding circuits.

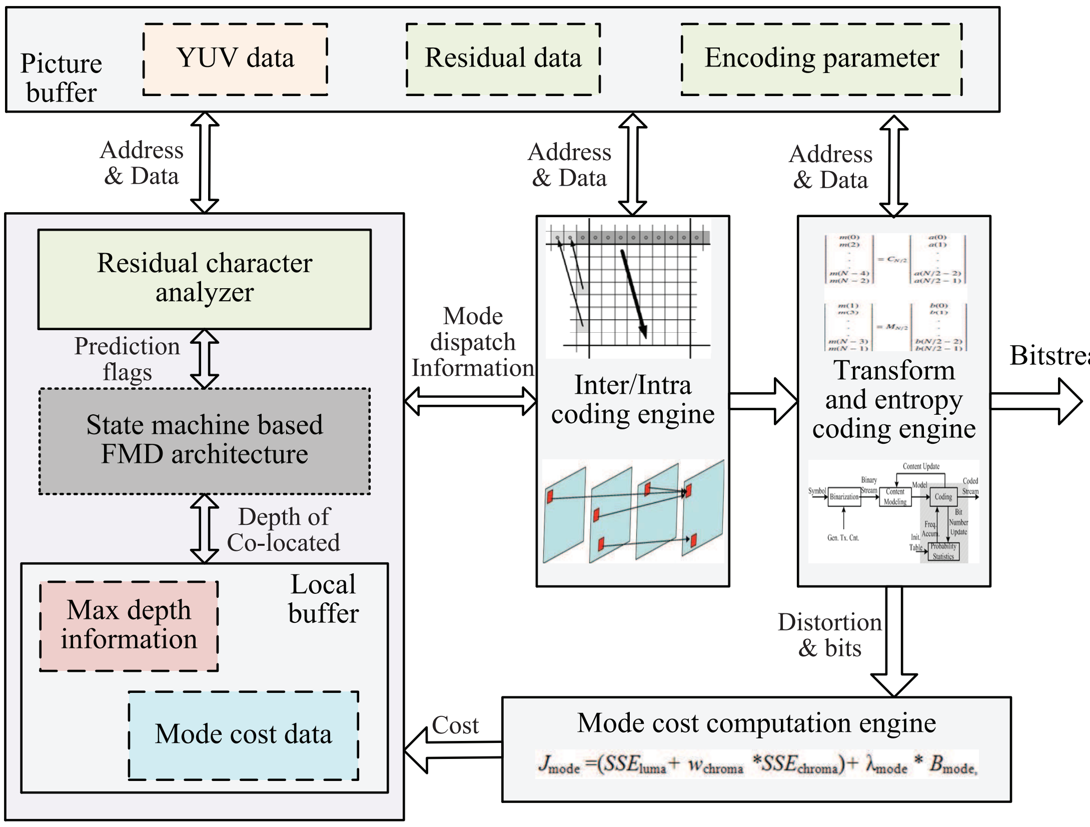
Picture buffer
YUV data
Residual data
Encoding parameter
Address & Data
Address & Data
Address & Data
Residual character analyzer
Prediction flags
State machine based FMD architecture
Depth of Co-located
Max depth information
Local buffer
Mode cost data
Mode dispatch Information
Inter/Intra coding engine
Transform and entropy coding engine
Bitstream
Distortion & bits
Cost
Mode cost computation engine
J_mode =(SSE_luma+ w_chroma *SSE_chroma)+ λ_mode * B_mode,

Figure 4.1: The framework of video codec where the mode dispatch module is embedded.

The picture buffer is used to store the original YUV data and residual data generated from the prediction coding process. The configuration parameters, e.g. picture order count (POC), Qp, frame size, *etc*, are also recorded in this buffer. Other modules can access these data according to the corresponding address using the standard read/write operation. In the inter/intra coding engine, the normative prediction coding processes stated in Section 2.3.1 are conducted following the instructions generated by our proposed mode dispatch module. By defining the interface in advance, inter/intra modules designed by others can also be easily integrated into this framework, to maintain the compatibility.

Then, the generated prediction residual will be transformed to the frequency domain and encoded in the following transform and entropy coding engine to form a standard bitstream. To decide the cost of a certain candidate mode, the distortion and the number of bits used to encode this mode are adopted in the mode cost computation engine.

Finally, the cost information and other information, such as the co-located maximum depth and the residual characters are employed in the state machine based fast mode dispatch module, which is used to dispatch the mode decision process. The output of this module is the candidate mode needs to be estimated by inter/intra prediction. Two surrounding circuits, the residual character analyzer and the local buffer are also designed to support the mode dispatch module.

As for the module of residual character analyzer, it is used to analyze the character of the generated residual by inter mode 2Nx2N, and to decide the conditions defined in Section 3.3.2. This part involves high computational complexity in order to analyze the residual, and we propose to implement this module in the on-chip processor using software algorithm as stated in Section 3.3.1 to decrease the consumed resources. For the depth prediction algorithm (Section 3.2), the maximum depth information of the previous frame should be stored for later usage in the prediction stage. Hence, a local buffer is also designed to store these information and the access to the local buffer is also based on the standard read/write operation.

In this section, we mainly focus on the hardware implementation of the state machine based mode dispatch module. The realization of the internal architecture of this module is depicted using block diagram in Figure 4.2.

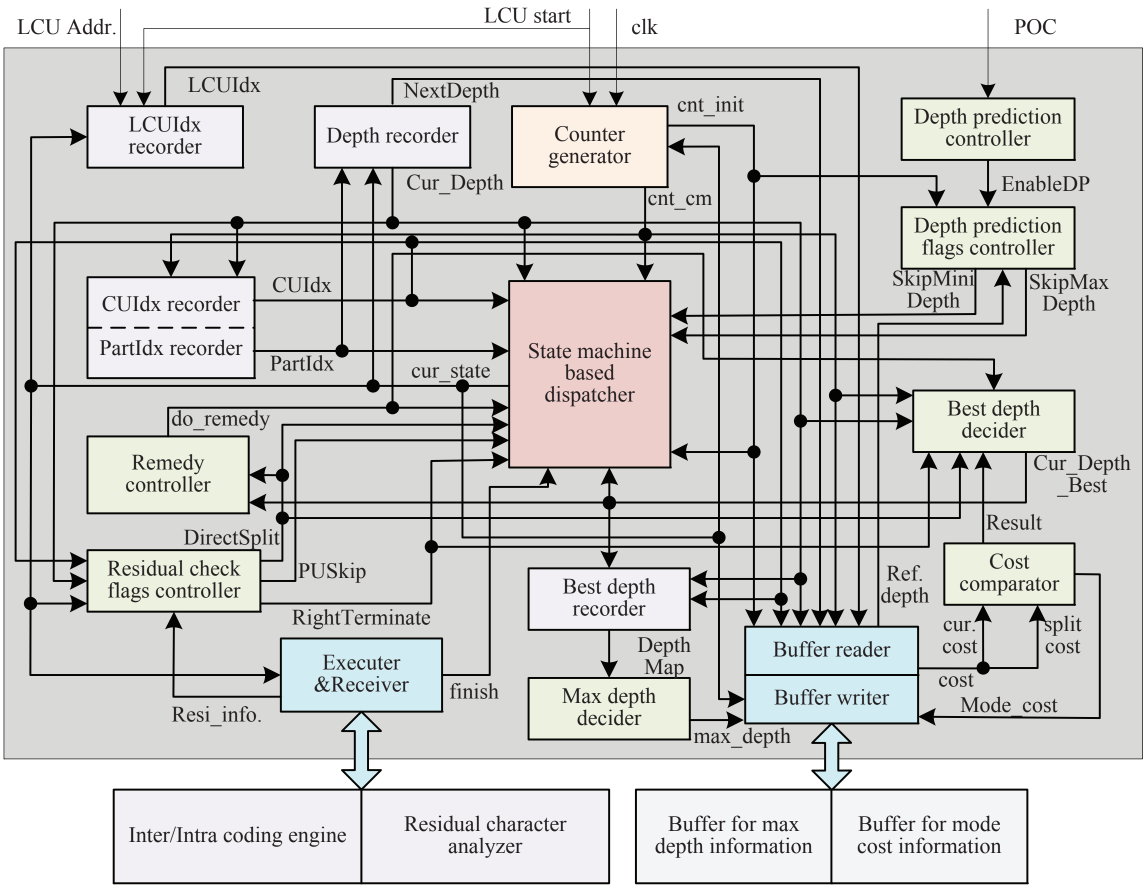

Figure 4.2: The proposed hardware architecture for the mode dispatch module.

In this mode dispatch module, mainly 4 kinds of elements are contained according to their functional definition: information recording element, controlling and decision making element, interface element, and the core mode dispatcher.

The information recording elements are LCUIdx recorder, depth recorder, CUIdx and PartIdx recorder, and best depth recorder. The LCUIdx recorder is used to keep track of the address of current LCU, and this address will be used as the address to access the maximum depth of the co-located LCU from previous frame and the address to save the maximum depth of current LCU at the end of the check of current LCU for later reference. For the depth recorder, the current depth and the next depth (the index of next depth is increased by 1) are recorded inside it. When all the candidate modes of current depth are all checked and it is decided to split, current depth will be increased by 1. On the other hand, when the checks of every 4 CUs of current depth are finished, current depth will be decreased by 1.

The index of each CU is generated in the CUIdx and PartIdx recorder to control the process flow among different CUs with different depthes. The method adopted to index each CU is illustrated in Figure 4.3. All the CUs belong to the same depth are indexed by CUIdx in an ascending way and every 4 CUs rooted in the same parent CU are indexed by PartIdx.

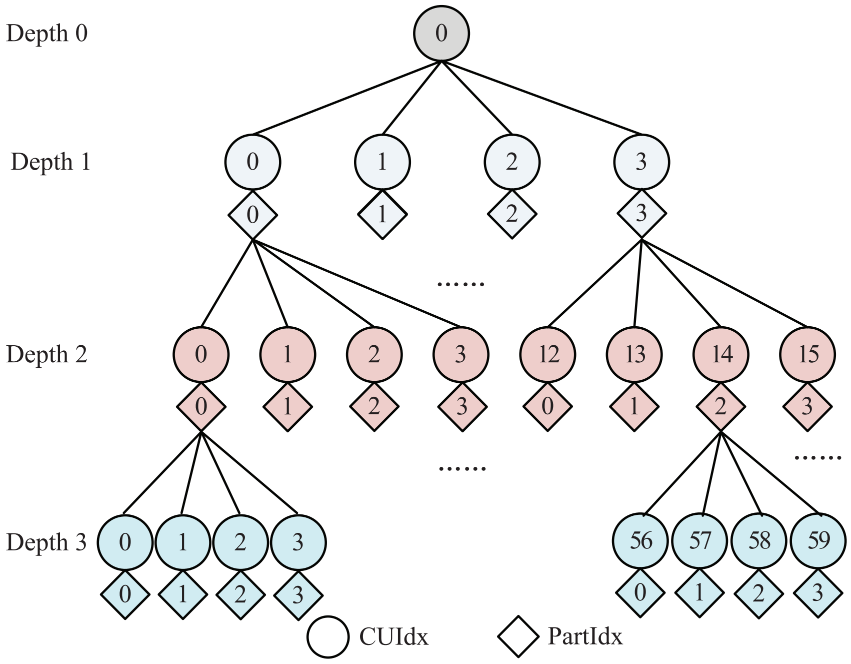

Figure 4.3: The index of the CUs.

The best depth recorder is used to record the division statuses of the CUs with depth 1 and 2. If a certain CU is decided to split, it will be marked as 0 using a 1 bit flag. Hence, 20 bits in total are needed to record the division information. The best depth recorder works together with the max depth decider, which is one of the controlling and decision making elements. From the best depth recorder, a depth map will be sent to the max depth decider to find the maximum depth of current LCU. For example, if all the CUs with depth 1 are marked as 1, the maximum depth is decided as 1. As depicted in Figure 4.4, the maximum depth is decided as 3.

The controlling and decision making elements are counter generator, cost comparator, depth prediction controller, depth prediction flags controller, residual check flags controller, remedy controller, best depth decider, and max depth decider. Two counters are generated in the counter generator: cnt_init and cnt_cm. The cnt_init is designed to control the flow of reading depth information from the local buffer. The cnt_cm is adopted to control the process of fetching the cost data of current CU and split cost of 4 sub-CUs in the next depth. Under the control of cnt_cm, the cost information of these 5 CUs is loaded to the cost comparator one by one, and then, the cost comparator will do a comparison between the cost of current depth and the overall split cost of next depth to find out the smaller one.

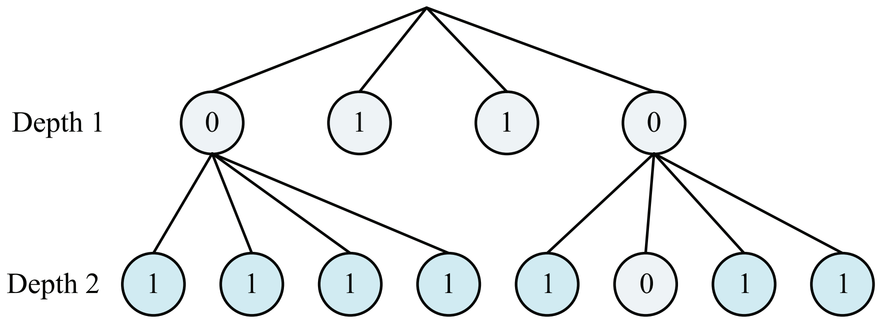

Figure 4.4: An example of the depth map.

As stated in Section 3.2.3, the proposed depth prediction algorithm is periodically disabled to prevent error accumulation. Hence, the depth prediction controller serves to turn on/off the depth prediction according to the POC of current picture. The implementation logic of this controller is depicted in Fig. 4.5. When it is enabled, the depth prediction flags controller will set the flags (denoted "SkipMiniDepth" and "SkipMaxDepth") used to skip unnecessary depth on the basis of the maximum depth of co-located LCU.

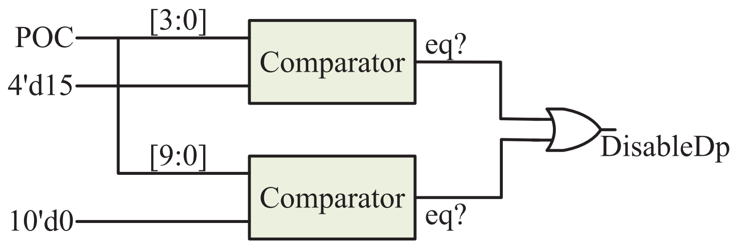

Figure 4.5: The implementation logic of the depth prediction controller.

As for the residual check flags controller, it is used to set the flags adopted to control the mode check flow according to the analysis results produced from the on-chip processor. These flags are "RightTerminate", "DirectSplit", and "PUSkip" corresponding to the conditions defined in Section 3.3.2 and the PU skip strategy stated in Section 3.3.3, respectively. For the "DirectSplit" flag, it is referred not only in the mode check of CU of current depth, but also used when the check over 4 sub-CUs of next depth is finished. Consequentially, this flag is stored for each CU during the whole check process of current LCU, and it is recorded in a series of registers according to the depth and the index of current CU. This saved information will be loaded to the remedy controller to decide whether remedy process is needed in line with the discussion in Section 3.3.4.

In the best depth decider, current depth will be judged to determine whether it is better than the next depth. If one of the following conditions is satisfied, current depth can be recognized as the best depth of current CU.

(1) The cost of current depth is smaller than the overall split cost of next depth;
(2) The "RightTerminate" flag is true. That is, no check is performed in the next depth;
(3) Current depth is the maximum depth of 3 to be exactly as defined;
(4) The "DirectSplit" flag is true and the remedy process is needed.

The executer & receiver and buffer reader & writer are interface elements. The executer & receiver serves as the interface between the proposed module and the inter/intra prediction engine as well as residual character analyzer. Through this interface, the instruction and the data are sent outside. The inter/intra prediction engine or the residual character analyzer will start when receiving the instruction. The data denotes the mode which needs to be checked. The complete signal and residual analysis results are received in from outside. This interface works on the basis of simple handshake protocol shown in Figure 4.6.

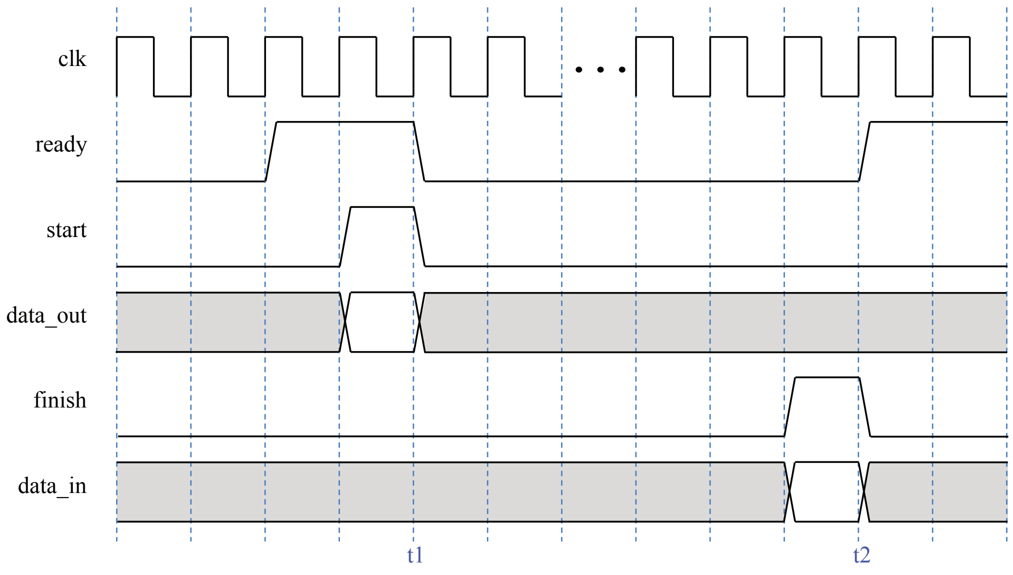

Figure 4.6: The simple handshake based protocol for the executer & receiver.

In Figure 4.6, signals “ready”, “finish”, and “data_in” are input, while signals “start” and “data_out” are output, relative to the proposed module. Signals “ready” and “finish” denote that the inter/intra prediction engine as well as residual character analyzer have been prepared well and complete the corresponding process, respectively. In this time chart, only when both “ready” and “start” are true, i.e. $t1$, the data transaction between proposed module and outside modules is recognized as successful. On the other hand, $t2$ denotes that the processing results from outsider modules have been received successfully.

According to above analysis, the communication and data transaction between the proposed module and the inter/intra prediction engine can be established using only three signals. This simply protocol can work effectively and reduce the number of interface signals.

The buffer reader & writer is designed as the interface between the proposed module and the local buffer. The maximum depth information of co-located LCU and cost data are loaded into the proposed module according to the standard ram read/write protocol, as shown in Figure 4.7. The maximum depth information is stored in accordance with the index of LCU, while cost data is saved in line with the depth and the CU index.

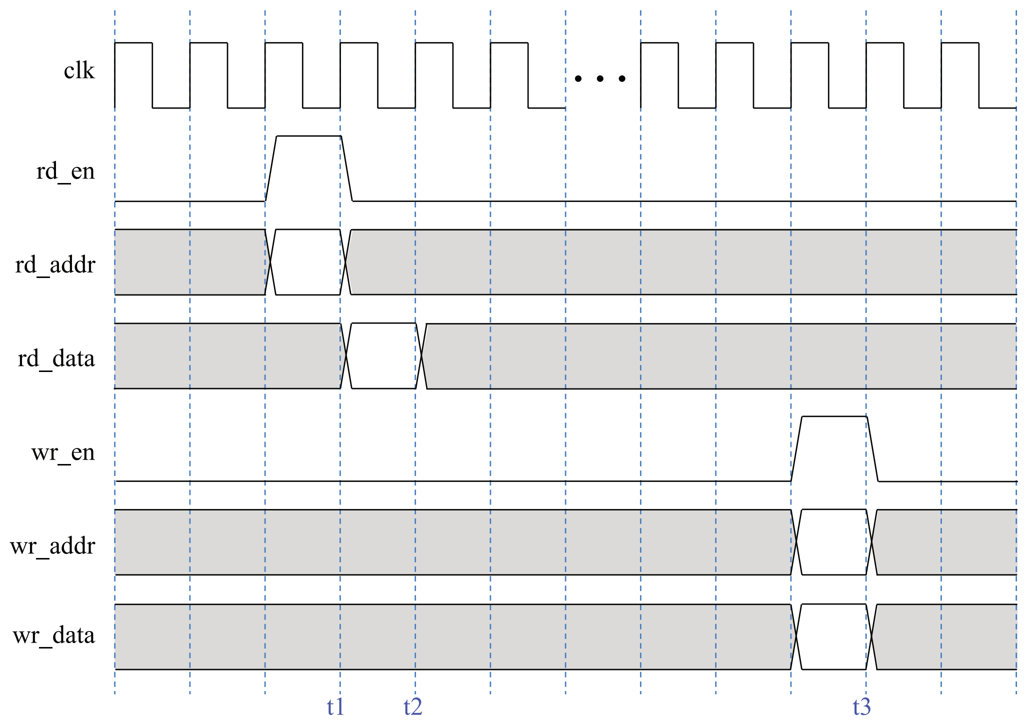

Figure 4.7: The standard read/write protocol for the buffer reader & writer.

4.2.2 Proposed state machine based fast mode decision architecture

In this section, the realization of the state machine based dispatcher is discussed. In this work, 13 states are defined: INITIAL, INTER_2Nx2N, INTER_2NxN, INTER_Nx2N, INTER_2NxnU, INTER_2NxnD, INTER_nLx2N, INTER_nRx2N, INTRA_2Nx2N, RESI_CHECK, SAVE_SPLIT, CHECK_MODE, and SAVE_MAX_DEPTH.

In the INITIAL state, the maximum depth data of the co-located LCU is loaded into the mode dispatch module under the control of counter cnt_init, and according to this information, the value of the flags "SkipMiniDepth" and "SkipMaxDepth" are decided. The states from INTER_2Nx2N to INTRA_2Nx2N denote the check process of the corresponding mode.

The analysis process of the character of the residual is conducted in the state RESI_CHECK. In the SAVE_SPLIT state, the intermediate information of current depth is saved, and the mode check process moves on to the next depth.

For the CHECK_MODE state, first, the cost information is read in; then, the cost of current depth competes with the overall split cost of next depth, and the smaller one is written out to the local buffer, controlled by cnt_cm. Also, the index of CU is updated during this state.

Finally, in the state SAVE_MAX_ DEPTH, the maximum depth of current LCU is sent to the local buffer, for later reference by next frame.

Four kinds of state transition are defined according to the fast decision algorithms. First of all, the normal state transition without using fast algorithms is illustrated in Figure 4.8.

After a new LCU check starts, all the candidate modes of current depth will be checked sequentially, and when all these modes are checked (this condition is abbreviated as "ACMC"),

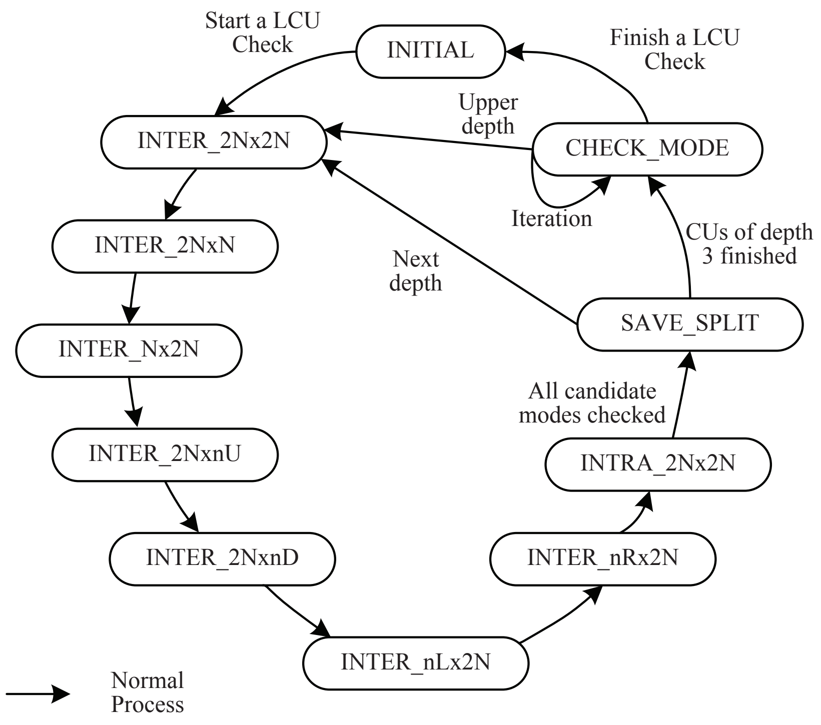

Figure 4.8: The normal state transition diagram.

the state will be set as SAVE_SPLIT. If current depth is not equal to the maximum depth (indexed by 3), the check of next depth will start, and the next state will be set as INTER_2Nx2N. On the other hand, if current depth is equal to 3 and all four CUs rooted in the same parent CU are checked (this condition is abbreviated as “CD3F”), next state will be CHECK_MODE.

In CHECK_MODE, current depth will be assigned as the value of its upper depth, and then by comparing the costs, whether this node (in the view point of quadtree structure) needs to split will be determined. If this node is the fourth one of its parent node, current depth will be upgraded to the upper level again to decide whether to split this parent node. This process is marked as “iteration” in Figure 4.8. Conversely, if this node is not the fourth one of its parent node, its neighboring CU of the same depth will be check, INTER_2Nx2N will be the next state. When all the CUs of all depths are checked, the check of this LCU will be finished.

When the depth prediction algorithm is enabled, the corresponding state transition is depicted in Figure 4.9.

If “SkipMiniDepth” is true, the check of depth 0 will be skipped, and next state will be set as SAVE_SPLIT. Otherwise, the same check process as stated above will be activated.

When current depth is equal to 2 and the flag “SkipMaxDepth” is true, the maximum depth will not be checked, and CHECK_MODE will be next state, as shown in Figure 4.9. When all the CUs of all depths are checked, the maximum depth of current LCU will be saved in the SAVE_MAX_DEPTH state.

Figure 4.10 demonstrates the state transition when the residual check algorithm is adopted. The INTER_2Nx2N will be first conducted, and then during RESI_CHECK, the generated residual will be analyzed in an on-chip processer. According to the analysis results, the values of the flags will be determined. If flag “RightTerminate” is true, mode check processes of the

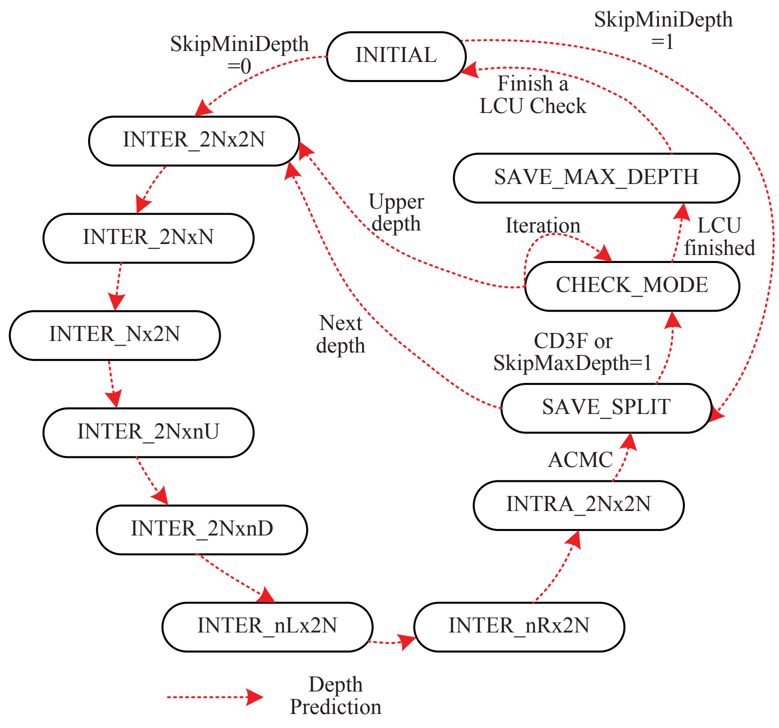

Figure 4.9: The state transition diagram for depth prediction algorithm.

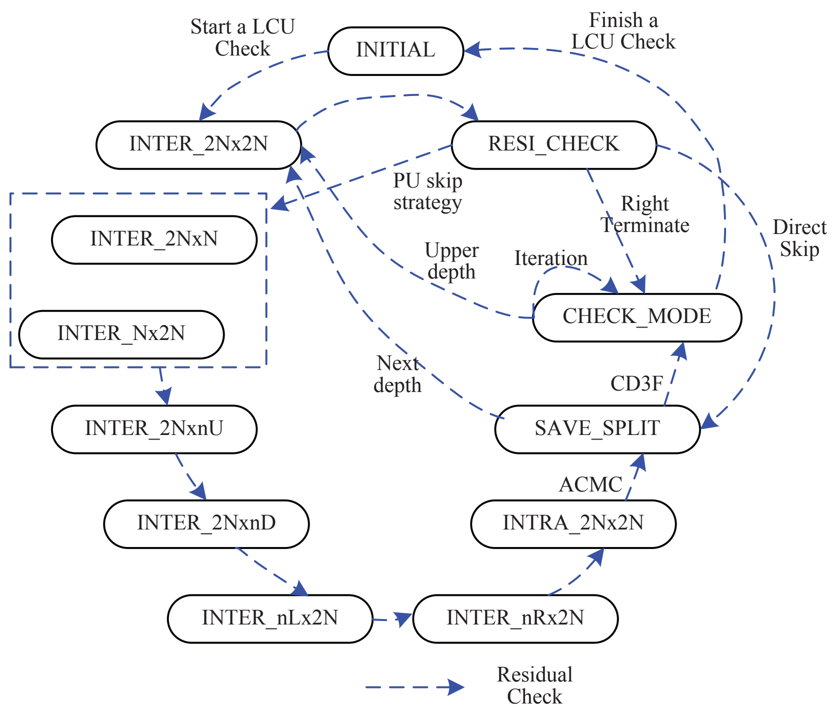

Figure 4.10: The state transition diagram for residual check algorithm.

following depths will be skipped, and CHECK_MODE will be set as next state. Then, if "DirectSplit" flag is true, check of current depth candidates will be unnecessary, and mode check process will carry on to the next depth, so next state will be SAVE_SPLIT. Finally, when "PUSkip" flag is set as true, one or both of the INTER_2NxN and INTER_Nx2N will be checked according to the discussion presented in Section 3.3.3. Otherwise, the same check process as the normal process in Figure 4.8 will be conducted.

After comparing the costs of current depth and next depth in CHECK_MODE, whether a remedy processes is needed when "DirectSplit" flag is true will be decided. If current depth is still better, the remedy process will be activated, and INTRA_2Nx2N will be checked first as stated in Section 3.3.4. If INTRA_2Nx2N is found to be a better candidate, other PU candidates will be discarded and the mode decision process will go back to the state CHECK_MODE. On the contrary, all the 6 PU candidate modes will be check sequentially to find out the best coding mode for current depth. The above stated state transition for remedy process is depicted in Figure 4.11.

Combine the above-mentioned four kinds of state transition together, the overall state transition diagram is illustrated is Figure 4.12. The corresponding state transition conditions are described in tabular form, shown in Table 4.1.

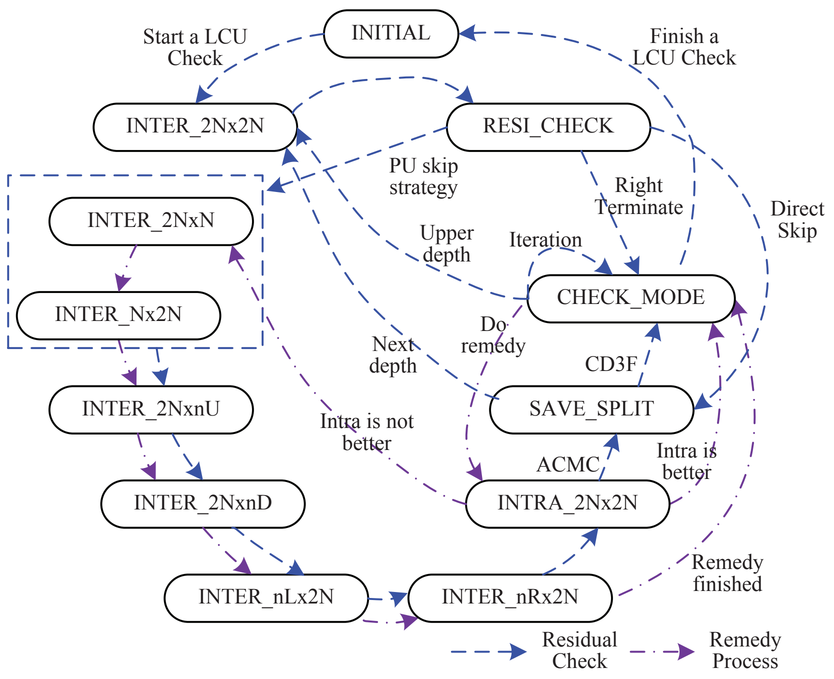

Figure 4.11: The state transition diagram for remedy process.

The number of states checked in each transition is also estimated and the results are given in Table 4.2. The minimum and maximum number of states are evaluated on the basis of a whole LCU for each condition. As for the common transition, 10 states are checked for each of 21 CUs indexed by depth 0, 1, 2, and 9 states are checked for each of 64 CUs indexed by depth 3. With the addition of one Initial state, there are totally 787 states needed to be checked. When SkipMiniDepth is true, 8 states can be skipped. With the addition of one SAVE_MAX_DEPTH state, there are totally 780 states needed to be checked. Although only 8 states are skipped, but all these states are conducted on the basis of LCU and hence, the

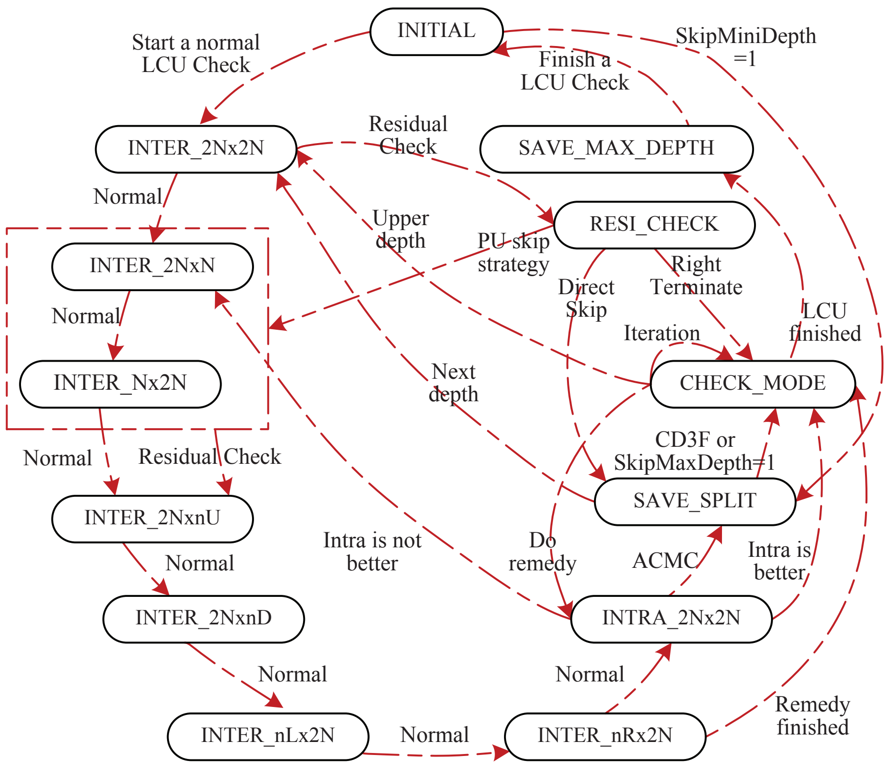

Figure 4.12: The overall state transition diagram.

time saving is considerable. On the other hand, when SkipMaxDepth is true, all the states of CUs in depth 3 are skipped.

When residual check is enabled, sometimes only 4 states need to be checked, while sometimes, the total number of states are more than that of common transition. This is because, the RESI_CHECK state will always be checked for all the 85 CUs in a whole LCU. For example, the minimum number of states when PUSkip is true is the same with that common transition. However, under this situation, time saving can also be obtained. This is because the time consumed by the extra RESI_CHECK state is relatively smaller than the time saved from skipping some states, such as INTER_2NxN or INTER_Nx2N.

4.3 Hardware implementation of the proposed fast DCD algorithm

As is known, in most natural video sequences, the pixels of a certain block are always tending to change along a certain direction. This direction can be called dominant direction. Hence, if the dominant direction of a block can be estimated in advance, the candidate modes whose directions are close to this direction can be chosen as the candidates, while others will be skipped without being checked.

In previous works [27, 29, 30], two common methods were adopted to detect the dominant direction: Sobel operator and extensive pixel based search.

According to the theory of Sobel operator, the dominant direction is estimated as following.

Table 4.1: The overall state transition.

Current state	Conditions	Next state
INITIAL	*(SkipMiniDepth=1) & (cur_depth=0)*	SAVE_SPLIT
	Otherwise	INTER_2NxnU
INTER_2Nx2N	*Always*	RESI_CHECK
INTER_2NxN	*PU skip strategy*	INTER_2NxnU
	Otherwise	INTER_Nx2N
INTER_Nx2N	*PU skip strategy*	INTER_2NxN
	Otherwise	INTER_2NxnU
INTER_2NxnU	*Always*	INTER_2NxnD
INTER_2NxnD	*Always*	INTER_nLx2N
INTER_nLx2N	*Always*	INTER_nRx2N
INTER_nRx2N	*Finish remedy process*	CHECK_MODE
	Otherwise	INTRA_2Nx2N
INTRA_2Nx2N	*(Remedy process needed) & (Intra is better)*	CHECK_MODE
	(Remedy process needed) & (Intra is not better)	INTER_2NxN
	Otherwise	SAVE_SPLIT
SAVE_SPLIT	*(SkipMaxDepth=1) & (cur_depth=2)*	CHECK_MODE
	cur_depth=3	CHECK_MODE
	Otherwise	INTER_2Nx2N
CHECK_MODE	*All 4 CUs rooted in same parent CU are checked*	CHECK_MODE
	Remedy process needed	INTRA_2Nx2N
	All CUs of all depths are checked	SAVE_MAX_DEPTH
	Otherwise	INTER_2Nx2N
RESI_CHECK	*DirectSplit=1*	SAVE_SPLIT
	RightTerminate=1	CHECK_MODE
	Check 2NxN and not Nx2N	INTER_2NxN
	Check Nx2N and not 2NxN	INTER_Nx2N
	Otherwise	INTER_2NxN
SAVE_MAX_DEPTH	*Always*	INITIAL

Table 4.2: The number of states checked in each transition.

Transition	Condition	Minimum	Maximum
Common		787	787
Depth prediction	SkipMiniDepth	780	780
	SkipMaxDepth	212	212
Residual check	RightTerminate	4	864
	PUSkip	787	872
	DirectSplit	746	872

Generally, Sobel operator adopts two convolution kernels which respond to the degree of the differences in vertical and horizontal directions, respectively. These kernels are called Sobel masks, as shown in Figure 4.13.

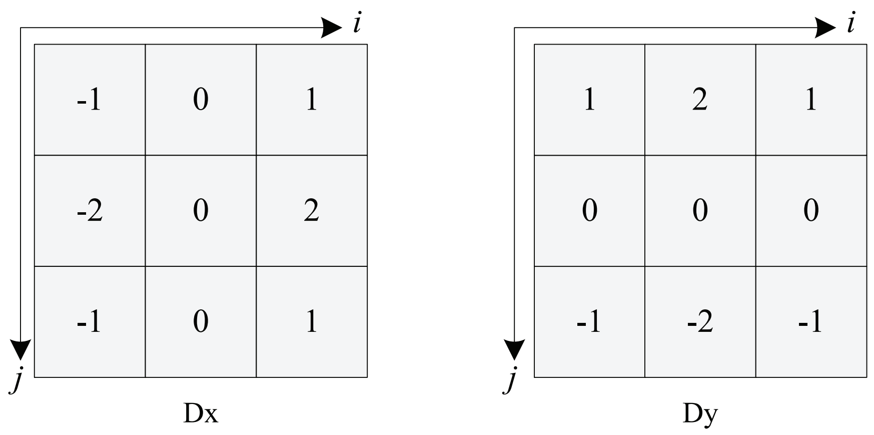

Figure 4.13: Sobel masks for gradients calculation.

For a certain pixel $p_{i,j}$, in a block of a frame, its corresponding edge vector, $\overrightarrow{D}_{i,j} = \{dx_{i,j}, dy_{i,j}\}$, is defined as

$$\begin{aligned} dx_{i,j} &= p_{i+1,j-1} + 2 \times p_{i+1,j} + p_{i+1,j+1} \\ &- p_{i-1,j-1} - 2 \times p_{i-1,j} - p_{i-1,j+1} \end{aligned} , \tag{4.1}$$

$$\begin{aligned} dy_{i,j} &= p_{i-1,j-1} + 2 \times p_{i,j-1} + p_{i+1,j-1} \\ &- p_{i-1,j+1} - 2 \times p_{i,j+1} - p_{i+1,j+1} \end{aligned} , \tag{4.2}$$

where $dx_{i,j}$ and $dy_{i,j}$ stand for the degree of the differences in vertical and horizontal directions, and the amplitude of the edge vector can be evaluated by

$$Amp\left(\overrightarrow{D}_{i,j}\right) = \left|dx_{i,j}\right| + \left|dy_{i,j}\right| . \tag{4.3}$$

Finally, the gradient of the pixel $p_{i,j}$, i.e. the estimated direction, is calculated as

$$Ang\left(\overrightarrow{D}_{i,j}\right) = \arctan\left(\frac{dy_{i,j}}{dx_{i,j}}\right) . \tag{4.4}$$

When the gradients of all the pixels in a certain block are calculated, a statistics histogram is established to decide the dominant direction of this block. In order to achieve this, the amplitudes of the edge vectors of the pixels with similar direction are summed up. The one with the highest amplitude can be assumed as dominant direction.

For the extensive pixel based search algorithm, following strategy is adopted to find the dominant direction.

As is known, there are 8 directional candidates defined in H.264/AVC, as shown in Figure 4.14. The extensive pixel based search algorithm proposes to calculate all the differences between two neighboring pixels located along all these predefined directions, and use the differences to estimate the strengths of these directions. These pixel directional strengths are expressed as

$$d_m\left(Mode0\right) = \left|p_{i,j+1} - p_{i,j}\right| , \tag{4.5}$$

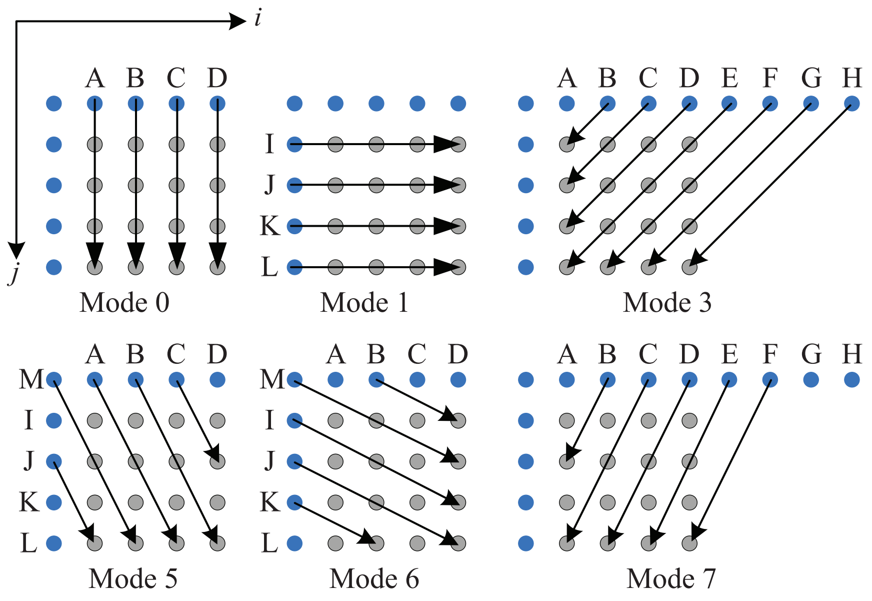

Figure 4.14: The directional intra modes in H.264/AVC.

$$d_m\left(Mode1\right) = \left|p_{i+1,j} - p_{i,j}\right| \ , \tag{4.6}$$

$$d_m\left(Mode3\right) = \left|p_{i-1,j+1} - p_{i,j}\right| \ , \tag{4.7}$$

$$d_m\left(Mode4\right) = \left|p_{i+1,j+1} - p_{i,j}\right| \ , \tag{4.8}$$

$$d_m\left(Mode5\right) = \left|p_{i+1,j+2} - p_{i,j}\right| \ , \tag{4.9}$$

$$d_m\left(Mode6\right) = \left|p_{i+2,j+1} - p_{i,j}\right| \ , \tag{4.10}$$

$$d_m\left(Mode7\right) = \left|p_{i-1,j+1} - p_{i,j}\right| \ , \tag{4.11}$$

$$d_m\left(Mode8\right) = \left|p_{i+2,j-1} - p_{i,j}\right| \ , \tag{4.12}$$

where i and j represent the horizontal and vertical positions of the pixel $p_{i,j}$. Then, the all the differences along one certain direction are summed up and averaged to form the strength of this direction. They are formulated as

$$D\left(Mode0\right) = \left(\sum_{m=1}^{12} d_m\left(Mode0\right)\right)/12 \ , \tag{4.13}$$

$$D\,(Mode1) = \left(\sum_{m=1}^{12} d_m\,(Mode1)\right)/12\,, \tag{4.14}$$

$$D\,(Mode3) = \left(\sum_{m=1}^{9} d_m\,(Mode3)\right)/9\,, \tag{4.15}$$

$$D\,(Mode4) = \left(\sum_{m=1}^{9} d_m\,(Mode4)\right)/9\,, \tag{4.16}$$

$$D\,(Mode5) = \left(\sum_{m=1}^{6} d_m\,(Mode5)\right)/6\,, \tag{4.17}$$

$$D\,(Mode6) = \left(\sum_{m=1}^{6} d_m\,(Mode6)\right)/6\,, \tag{4.18}$$

$$D\,(Mode7) = \left(\sum_{m=1}^{6} d_m\,(Mode7)\right)/6\,, \tag{4.19}$$

$$D\,(Mode8) = \left(\sum_{m=1}^{6} d_m\,(Mode9)\right)/6\,. \tag{4.20}$$

Finally, the direction which is estimated with lowest strength will be chosen as the dominant direction.

In this work, the above-mentioned two previous algorithms and the proposed fast DCD algorithm (proposed in Section 3.4.1) are implemented on the hardware platform to make an evaluation about the consumed hardware resources.

4.4 Synthesis results

4.4.1 Synthesis results for the proposed mode dispatch module

In this section, the synthesis results of the proposed mode dispatch module are first provided. This proposed architecture is described by using Verilog HDL and synthesized on the FPGA platform, and the results are given in Table 4.3

From the table, we can see that the proposed architecture achieves a maximum frequency as around 193 MHz with less than 1% of the total resources consumed.

Due to the fact that the proposed mode dispatch module is mainly used to decide the next candidate mode needed to be checked, the actual throughput of this module primarily depends on that of the inter/intra coding engine. However, we can make an estimation about the throughput by comparing with a common codec without using FMD algorithms. For a certain codec, the architecture of it is fixed, and the time consumed by checking each mode is also fixed. According to Chapter 3, it can be found that in order to find the best mode for a certain CU, only 30% of the time is consumed. That is, with the same hardware resources, more

Table 4.3: Synthesis results of the proposed mode dispatch module on FPGA.

FPGA	Cyclone IV GX
Device	EP4CGX150DF31I7
Logic element	334 / 149,760 (< 1 %)
- Combinational Functions	323 / 149,760 (< 1%)
- Total registers	204 / 149,760 (< 1 %)
Slow 1200mV 100C Model	193.65 MHz
Slow 1200mV -40C Model	218.67 MHz

Table 4.4: Total memory bits consumed by the proposed local buffer on FPGA.

Variable	Bits	Number	Total bits	Overall bits
Max. depth	2	2048	4096	6784 / 6635520
Mode cost	32	84	2688	($\approx$ 1 %)

modes can be checked compared with the a codec without FMD. Therefore, the proposed module can improve the overall throughput of the codec by almost three times.

Moreover, when maintaining the same throughput as a codec without FMD, the overall hardware resources can be reduced by using the proposed mode dispatch module. Now, we consider a certain codec with a frequency of 100MHz. A throughput of 377.48 Msamlpes/s is required when processing 4Kx2K (4096x2048, 30fps, 4:2:0 format) video sequences. When FMD algorithms are not used, to achieve this requirement, we can design three inter/intra coding engines working in parallel and the throughput of each engine is set as 126 Msamlpes/s. However, when using the proposed mode dispatch module, only one inter/intra coding engine with a throughput of 126 Msamlpes/s can satisfy the above requirement. This is because the proposed module can triple the overall throughout as discussed above. Therefore, the overall hardware resources are reduced for a certain codec containing the proposed mode dispatch module.

Finally, the total memory bits consumed by the proposed local buffer are evaluated and the results are tabulated in Table 4.4. The maximum depth of current LCU is represented by using a 2 bits register (e.g., $(01)_2 == depth\ 1$). In this work, 4Kx2K video sequence is targeted. Hence, there are 2048 ($(4096 \times 2048)/(64 \times 64)$) LCUs inside each frame and 4096 bits are consumed to save the maximum depth of all LCUs of one frame. To maintain a high accuracy, the cost of the best mode of current CU is represented by using a 32 bits variable. As stated, the mode cost information will be used throughout the mode decision process of a LCU. Therefore, the mode costs of all CUs of one LCU need to be stored and there are 84 ($4(depth\ 1) + 16(depth\ 2) + 64(depth\ 3)$) CUs inside one LCU. Hence, 2688 bits are consumed to save the mode cost information. The overall bits consumed by the proposed local buffer is approximate 1% of the total memory bits of the selected FPGA in Table 4.3.

4.4.2 Synthesis results for the proposed fast DCD algorithm

The complexity of the proposed fast intra mode decision algorithm is compared with that of the previous algorithms, stated in above section. The complexities of these algorithms are estimated on the basis of the number of arithmetic operations used to derive the dominant direction. The evaluation is conducted on the basis of 4x4 block. Results are shown in Table 4.5.

Table 4.5: Number of arithmetic operations used to derive the dominant direction.

Method	arithmetic operations			
	Addition	Subtraction	Multiplication	abs()
Proposed	4	8	0	8
Sobel Operator [64]	95	96	64	32
Extensive Detection [29]	38	42	0	42

To perform a further comparison, the three algorithms listed in Table 4.5, are described by using Verilog and synthesized on the FPGA platform. The results are given in Tables 4.6 and 4.7.

Table 4.6: Synthesis results of the three fast intra mode decision algorithms on FPGA.

Algorithm	Proposed	Extensive Detection [29]	Sobel Operator [64]
FPGA	Cyclone IV E		
Device	EP4CE115F29I8L		
Circuit type	Combinational		
Logic element	316	1954	2856

Table 4.7: Estimated throughput of the proposed fast DCD algorithm.

Synthesis condition	Frequency (MHz)	Throughput(MSamples/s)
Slow 1000mV 100C Model	139.1	2225.6
Slow 1000mV -40C Model	146.7	2347.2

According to Table 4.5, the proposed DCD algorithm achieves a significantly lower complexity, i.e., when implemented on the hardware platform, it consumes the fewest hardware resources compared with the other two works. This is illustrated in Table 4.6, where the proposed DCD algorithm consumes only 1/6 and 1/9 of the resources of the two previous works, respectively. Moreover, the simple calculation pattern leads to easy hardware implementation.

As shown in Table 4.7, the maximum working frequency is higher than 139.1 MHz, and the corresponding estimated throughput is about 2225.6 MSamples/s. This demonstrates that it can support the real-time processing of 4Kx2K (4096x2048, 30fps, 4:2:0 format, 377.5 MSamples/s) video sequences.

Finally, the performance of the proposed fast mode decision algorithm is compared with that of several previous works, listed in [64, 65]. The performance comparison results are given in Table 4.8. It can be seen from this table that our algorithm achieves the same time reduction while maintains a lower bitrate loss, compared with the work in [65].

When compared with previous work in [64], this fast intra decision algorithm achieves a two times more complexity reduction, at the cost of certain degree of bitrate increase. However, combined with Table 4.5 and 4.6, our low complexity algorithm can save more than 88.9% resources when implemented on the hardware platform.

Table 4.8: Performance of fast intra decision algorithm compared with previous works.

Method	Proposed	Ref. [65]	Ref. [64]
	(ΔBR,ΔT)	(ΔBR,ΔT)	(ΔBR,ΔT)
Class A	(2.71, -52.4)	(5.85, -52.8)	(0.55, -20.4)
Class B	(1.75, -53.9)	(6.65, -55.5)	(0.70, -20.5)
Class C	(2.00, -51.4)	(3.50, -56.1)	(0.74, -19.4)
Class D	(2.31, -50.5)	(4.35, -62.5)	(0.94, -19.5)
Class E	(4.19, -58.5)	(-, -)	(0.84, -20.1)
AVG	(2.59, -53.3)	(5.09, -56.7)	(0.75, -20.0)

4.5 Conclusion

In this chapter, the hardware architectures of the fast mode decision algorithms were discussed. First, a state machine based module for the depth prediction combined with the residual check algorithm was presented, and the corresponding system framework as well as the realization of the internal architecture of this module were analyzed in detail. Hardware synthesis results demonstrated that the proposed architecture achieved a maximum frequency of about 193 MHz.

Then, the hardware implementation of the proposed fast DCD algorithm along with two previous works was presented. The performance comparison results showed that this proposed algorithm achieved almost the same time reduction while maintained a lower bitrate loss compared with other work. Furthermore, our low complexity algorithm could save more than 88.9% resources when implemented on the hardware platform.

Chapter 5

High-performance transform architecture for H.265/HEVC

5.1 Introduction

As is stated, one key feature of the H.265/HEVC is supporting transform of different block sizes, ranging from 4x4 to 32x32. Therefore, the corresponding hardware architecture should also be flexible enough to enable the computation of all these sized transform. The existing designs for conventional transform of H.264/AVC can only support transform sizes of 4x4 and 8x8, resulting that they cannot be reused directly by H.265/HEVC. Taking account of this issue, we have investigated the possibility of the implementation of the transform for HEVC under the context of resource requirement and reusability. From this point of view, a hardware-oriented algorithm is devised for hardware implementation. A low cost and high throughput transform architecture has been designed for 1-D and 2-D transforms for HEVC [121, 122].

To achieve such objectives, some simplification strategies are adopted during the implementation, such as reusing part of the structure of larger sized transform by smaller size, and turning multiplication by constant into shift and sum operations. Moreover, a high-performance transposition memory is proposed to store and transpose the intermediate data between the 1-D and 2-D transforms. The transform architecture proposed in this chapter is implemented in the form of pipeline structure.

5.2 Algorithm for hardware implementation

5.2.1 Review of the transform of HEVC

The calculation of the 1-D transform of size NxN for HEVC is conducted as following. Let $R = [r(0), r(1), ..., r(N-1),]^T$ be an N-point input vector (a column of the residual matrix, to be exactly), and $M = [m(0), m(1), ..., m(N-1),]^T$ be the corresponding 1-D transform result. M is expressed as

$$M = C_N R, \quad N = 4, 8, 16, \textit{ and } 32, \tag{5.1}$$

where C_N denotes the N-point transform matrix. When $N = 4$, the 4-point 1-D transform is calculated as

$$\begin{bmatrix} m(0) \\ m(1) \\ m(2) \\ m(3) \end{bmatrix} = \begin{bmatrix} \mathrm{c0} & \mathrm{c0} & \mathrm{c0} & \mathrm{c0} \\ \mathrm{c1} & \mathrm{c2} & -\mathrm{c2} & -\mathrm{c1} \\ \mathrm{c0} & -\mathrm{c0} & -\mathrm{c0} & \mathrm{c0} \\ \mathrm{c2} & -\mathrm{c1} & \mathrm{c1} & -\mathrm{c2} \end{bmatrix} * \begin{bmatrix} r(0) \\ r(1) \\ r(2) \\ r(3) \end{bmatrix}. \tag{5.2}$$

The matrix calculation of Eq. (5.2) can be separated into two parts containing even- or odd-indexed components. This simplification is expressed as

$$\begin{bmatrix} m(0) \\ m(2) \end{bmatrix} = \begin{bmatrix} \mathrm{c0} & \mathrm{c0} \\ \mathrm{c0} & -\mathrm{c0} \end{bmatrix} * \begin{bmatrix} a(0) \\ a(1) \end{bmatrix}, \tag{5.3}$$

and

$$\begin{bmatrix} m(1) \\ m(3) \end{bmatrix} = \begin{bmatrix} \mathrm{c1} & \mathrm{c2} \\ \mathrm{c2} & -\mathrm{c1} \end{bmatrix} * \begin{bmatrix} b(0) \\ b(1) \end{bmatrix}, \tag{5.4}$$

where $a(0) = r(0)+r(3)$, $a(1) = r(1)+r(2)$, $b(0) = r(0)-r(3)$, and $b(1) = r(1)-r(2)$. Same as the case of 4-point transform, the computations of transforms of sizes 8x8, 16x16, and 32x32 can also be simplified to the separated calculations of even- and odd-indexed components. The detail is

$$\begin{bmatrix} m(0) \\ m(2) \\ . \\ . \\ . \\ m(N-4) \\ m(N-2) \end{bmatrix} = C_{N/2} \begin{bmatrix} a(0) \\ a(1) \\ . \\ . \\ . \\ a(N/2-2) \\ a(N/2-1) \end{bmatrix}, \tag{5.5}$$

and

$$\begin{bmatrix} m(1) \\ m(3) \\ . \\ . \\ . \\ m(N-3) \\ m(N-1) \end{bmatrix} = M_{N/2} \begin{bmatrix} b(0) \\ b(1) \\ . \\ . \\ . \\ b(N/2-2) \\ b(N/2-1) \end{bmatrix}, \tag{5.6}$$

where $a(i) = r(i) + r(N - i - 1)$, and $b(i) = r(i) - r(N - i - 1)$, for $i = 0, 1, ..., N/2 - 1$. $C_{N/2}$ denotes the N/2-point transform matrix, and $M_{N/2}$ is a matrix of size $(N/2) \times (N/2)$, whose entry on $(i, j)^{th}$ is defined as

$$m_{N/2}^{i,j} = c_N^{2i+1,j}, \quad 0 \le i, j \le N/2 - 1, \tag{5.7}$$

where $c_N^{2i+1,j}$ is the $(2i + 1, j)^{th}$ entry of C_N.

From Eq. (5.5) and Eq. (5.6), we can see that the N-point 1-D transform for HEVC can be computed by a partial butterfly approach using a (N/2)-point transform and a matrix-vector product of $(N/2) \times (N/2)$ matrix with an (N/2)-point vector. Moreover, Eq. (5.5) could be similarly further decomposed into $C_{N/4}$ and $M_{N/4}$, if $N = 16$ *or* 32. In this chapter, the direct implementation of DCT based on Eq. (5.5) and (5.6) is referred as the reference algorithm in the remainder of this chapter.

5.2.2 Hardware-oriented algorithm

Base on the analysis of above section, the 2-D forward transform in HEVC is defined as

$$D = CRC^T \ , \tag{5.8}$$

where CRC^T is called core 2-D transform and D, C, R denote the transformed coefficients, the transform matrix and the residual samples outputted from prediction coding process, respectively. This core 2-D transform contains two 1-D transforms.

In order to implement both the first and second 1-D transforms with a unique architecture, we proposed to separate the core 2-D transform as follows. For the first 1-D transform, it is calculated as

$$M = CR^T \ . \tag{5.9}$$

For the second 1-D transform, following computation is conducted,

$$D = CM^T \ . \tag{5.10}$$

On the basis of following matrix transformation,

$$D = CM^T = C(CR^T)^T = C(RC^T) = CRC^T \ , \tag{5.11}$$

we can see that the proposed separation of two 1-D transforms achieves the same results as the core 2-D transform.

According to [80], the coefficients of the transform matrixes are integer approximation of mathematical DCT matrixes, which are rounded to 8 bit integer accuracy including sign. These approximate matrixes are optimized with maximizing orthogonality. As stated, transforms of size 4x4 up to 32x32 are supported in HEVC. The transform matrixes used in 1-D 8x8 and 16x16 transform are given as follows.

$$\begin{vmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{vmatrix} , \tag{5.12}$$

$$\begin{vmatrix}
64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\
90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & -90 \\
89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\
87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\
83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\
80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\
75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\
70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\
64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\
57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\
50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\
43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\
36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 & 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\
25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\
18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\
9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 & 90 & -87 & 80 & -70 & 57 & -43 & 25 & -9
\end{vmatrix}. \tag{5.13}$$

It is easy to find out that the matrix coefficients C_{ij} $(i, j = 0...7)$ given in Eq. (5.12) have symmetry properties that are consistent with the DCT transform adopted in the previous standard:

(1) Even rows with indexes $0, 2, 4, 6$ have symmetric property with a symmetry point between the 3rd and 4th coefficients.
(2) Odd rows with indexes $1, 3, 5, 7$ have anti-symmetric property with an anti-symmetry point between the 3rd and 4th coefficients.
(3) Even rows with indexes $0, 4$ have additional symmetry points before the 2nd and 6th coefficients.
(4) Other even rows with indexes $2, 6$ have additional anti-symmetry points before the 2nd and 6th coefficients.

For transforms of larger sizes, the above mentioned symmetric (anti-symmetric) properties with symmetry points before the $(N/2)^{th}$ coefficients are repeated as well as the additional symmetry and anti-symmetry points before coefficients $N/4, 3N/4, n*N/8, n*N/16$ etc.

In the first step, we consider the calculation of 1-D transform

$$M_i = CR_i^T, \quad i = 0...7, \tag{5.14}$$

where M_i represents the ith column of intermediate matrix and R_i denotes the ith row of residual matrix. When using each row indexed by j $(j = 0...7)$ from transform matrix to multiply R_i, the sum results can be obtained as follows.

The calculation of even rows with indexes $0, 4$ can be summarized as the same group, and the results are

$$\begin{aligned} m_{0i} = &\ (((r_{i0} + r_{i7}) + (r_{i3} + r_{i4})) \\ + &\ ((r_{i1} + r_{i6}) + (r_{i2} + r_{i5}))) * 64, \end{aligned} \tag{5.15}$$

$$\begin{aligned} m_{4i} = &\ (((r_{i0} + r_{i7}) + (r_{i3} + r_{i4})) \\ - &\ ((r_{i1} + r_{i6}) + (r_{i2} + r_{i5}))) * 64. \end{aligned} \tag{5.16}$$

The calculation of even rows with indexes $2, 6$ can be also summarized as the same group, take 2nd row as an example and the result is

$$\begin{aligned} m_{2i} \quad = & \quad ((r_{i0} + r_{i7}) - (r_{i3} + r_{i4})) * 83 \\ + & \quad ((r_{i1} + r_{i6}) - (r_{i2} + r_{i5})) * 36, \end{aligned} \tag{5.17}$$

The calculation of odd rows with indexes $1, 3, 5, 7$ can also be summarized as the same group, take 1^{th} row as an example and the result is

$$\begin{aligned} m_{1i} \quad = & \quad (r_{i0} - r_{i7}) * 89 + (r_{i1} - r_{i6}) * 75 \\ + & \quad (r_{i2} - r_{i5}) * 50 + (r_{i3} - r_{i4}) * 18. \end{aligned} \tag{5.18}$$

Based on Eqs. (5.15 - 5.18), the calculation of 1-D 8x8 transform can be implemented through the flow-group depicted in Figure 5.1.

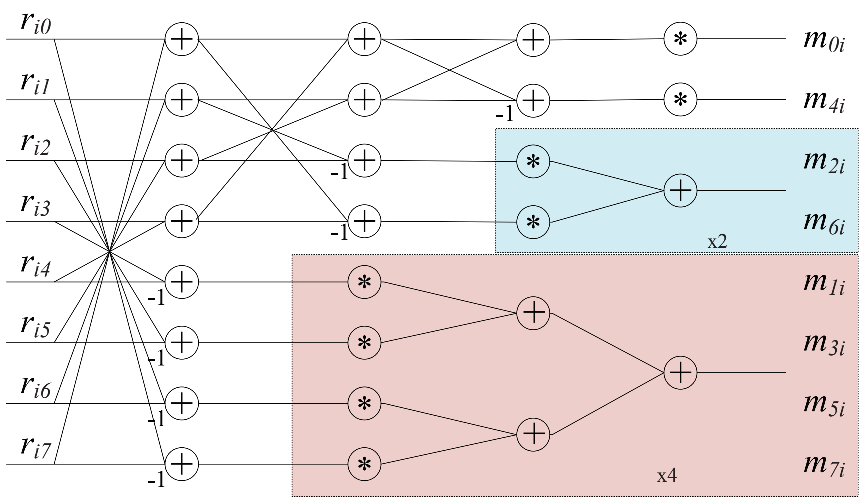

Figure 5.1: Flow-chart of the 1-D 8x8 transform.

The whole transform process is split into elementary parts, and each part operates with no more than two operands. The first calculation stage of this simplified 1-D forward transform is presented in Table 5.1.

Table 5.1: Operations in the first stage.

Input	Output
$r_{i0} + r_{i7}$	E_{i0}
$r_{i0} - r_{i7}$	O_{i0}
$r_{i1} + r_{i6}$	E_{i1}
$r_{i1} - r_{i6}$	O_{i1}
$r_{i2} + r_{i5}$	E_{i2}
$r_{i2} - r_{i5}$	O_{i2}
$r_{i3} + r_{i4}$	E_{i3}
$r_{i3} - r_{i4}$	O_{i3}

Table 5.2: Demonstration of the constants in multiplication converted into shift and sum.

Transform size	Row index	Coefficient	Shift Operation							Sum Operation
			≪6	≪5	≪4	≪3	≪2	≪1	≪0	
4x4	even	64	+							≪6
	odd	83	+		+			+	+	≪6+≪4+≪1+≪0
		36		+			+			≪5+≪2
8x8	odd	89	+		+	+			+	≪6+≪4+≪3+≪0
		75	+			+		+	+	≪6+≪3+≪1+≪0
		50		+	+			+		≪5+≪4+≪1
		18			+			+		≪4+≪1
16x16	odd	90	+		+	+		+		≪6+≪4+≪3+≪1
		87	+	+		-			-	≪6+≪5-≪3-≪0
		80	+		+					≪6+≪4
		70	+				+	+		≪6+≪2+≪1
		57		+	+	+			+	≪5+≪4+≪3+≪0
		43		+		+		+	+	≪5+≪3+≪1+≪0
		25			+	+			+	≪4+≪3+≪0
		9				+			+	≪3+≪0
32x32	odd	88	+		+	+				≪6+≪4+≪3
		85	+		+		+		+	≪6+≪4+≪2+≪0
		82	+		+			+		≪6+≪4+≪1
		78	+			+	+	+		≪6+≪3+≪2+≪1
		73	+			+			+	≪6+≪3+≪0
		67	+					+	+	≪6+≪1+≪0
		61	+					-	-	≪6-≪1-≪0
		54		+	+		+	+		≪5+≪4+≪2+≪1
		46		+		+	+	+		≪5+≪3+≪2+≪1
		38		+			+	+		≪5+≪2+≪1
		31		+					-	≪5-≪0
		22			+		+	+		≪4+≪2+≪1
		13				+	+		+	≪3+≪2+≪0
		4					+			≪2

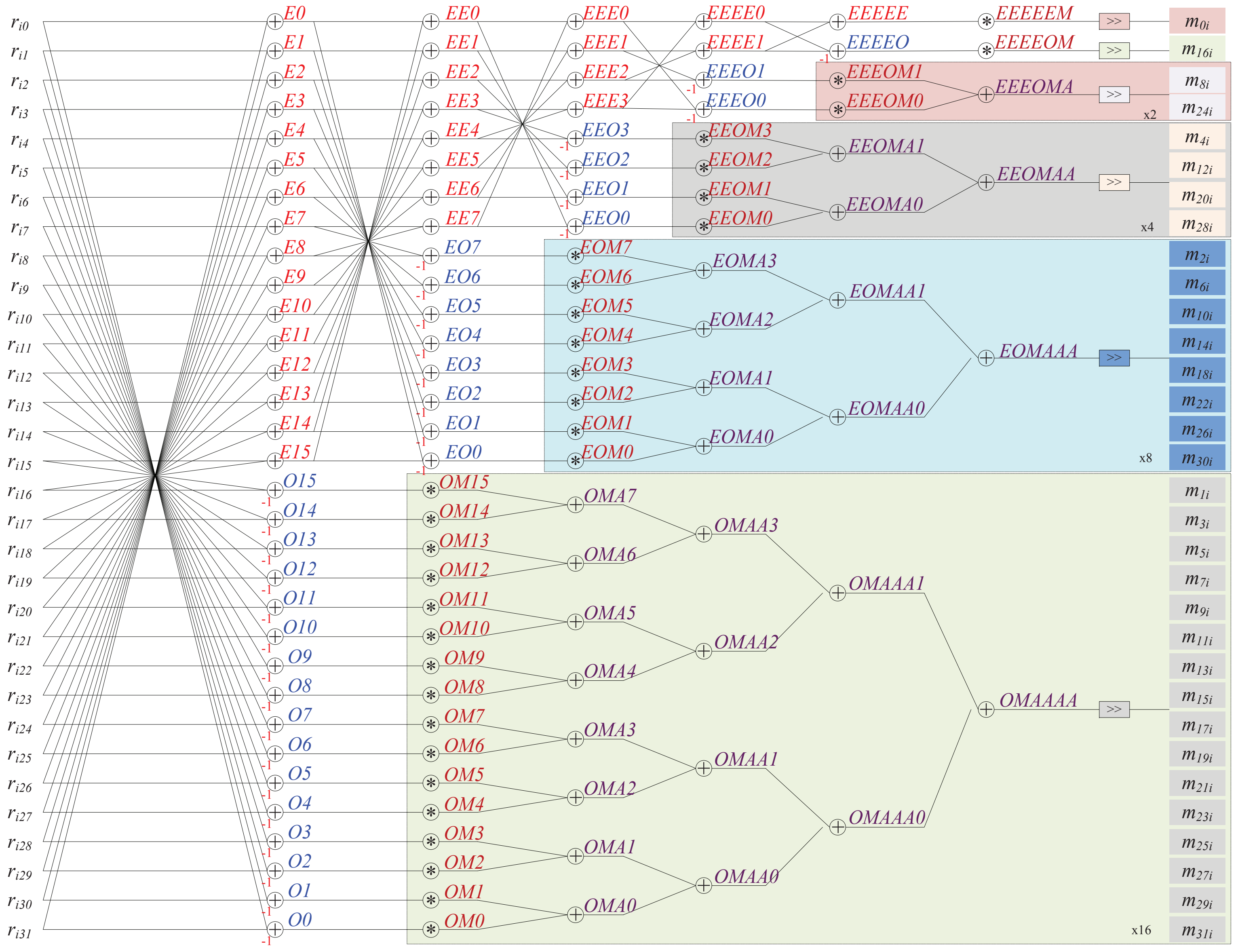

Figure 5.2: Flow-chart of the 1-D 32x32 transform.

In this process stage, only sums and subtractions are performed to the input. All the other following operations and corresponding calculation simplifications are conducted based on these results. In this table, r_{ij} ($j = 0, 1, ..., 7$) denote the input data from the i^{th} row of the residual matrix generated from the prediction stage in the encoder and E_{ij}, and O_{ij} ($j = 0, 1, ..., 3$) represent the output from the sum/subtraction operations.

When outputs from first stage are ready, the second process stage starts with different operations with respect to the row index. For even rows with indexes $0, 2, 4, 6$, the similar sum and subtraction processes are conducted based on the same rule as first stage. As for the odd rows with indexes $1, 3, 5, 7$, no other symmetric properties can be used, so multiplications between the outputs from first stage and the coefficients from the corresponding row of transform matrix are calculated.

As is known, the direct implementation of multiplication on hardware platform will lead to a relatively high resource cost. Therefore, for this kind of multiplied by constant operation, we propose to use shift and sum operations to produce the same result to decrease the hardware resource cost. Table 5.2 gives an explanation on how to convert multiplied by constant into shift and sum operations. The shift operation can be easily implemented by concatenating a number of zeros to the input data from the first stage.

Due to the fact that the coefficients in the same position of different rows with odd indexes in the transform matrix given by Eq. (5.12) are different, the multiplications have to be recalculated for each row. Therefore, for rows with indexes $1, 3, 5, 7$, the same part of the calculation flow chart is repeated four times in the flow-graph depicted in Figure. 5.1.

The third and fourth process stages are conducted by the same rules as stated above. Here, the same part of multiplication flow chart is repeated twice for the 2nd and 6th rows, while rows with indexes $0, 4$ do not need to repeat the multiplication operation based on the additional symmetry points before the 2nd and 6th coefficients. Moreover, the products of the each row are summed except for 0th and 4th rows.

After the first 1-D transform, the resulting transformed coefficients are scaled using right shift operation to 16 bit. This ensures a bit width of 16 after each transform stage.

Likewise, on the basis of similar derivation, the data flow of the transform with size 32x32 is depicted in Figure 5.2.

5.3 Hardware Architecture design

5.3.1 Hardware reuse

The proposed architecture design is presented based on the above-mentioned simplified algorithm. This architecture is described using Verilog HDL and implemented in the form of pipeline structure.

Moreover, the hardware of the proposed architecture can be easily reused by transforms of different sizes, which means that part of the structure of larger sized transform can be used by smaller sized transform without any change. In order to prove the above property, here we take a look at the transform matrix of size 4x4, shown as

$$\begin{vmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{vmatrix}. \tag{5.19}$$

It is easily found out that the coefficients of the 4x4 transform matrix also have symmetry properties that are consistent with that of 8x8 transform matrix:

(1) Even rows with indexes 0, 2 have symmetric property with a symmetry point between the 3rd and 4th coefficients.
(2) Odd rows with indexes 1, 3 have anti-symmetric property with an anti-symmetry point between the 3rd and 4th coefficients.

By the same way, we consider the calculation of first 1-D transform

$$M_i = C\widetilde{R}_i, \quad i = 0...3, \tag{5.20}$$

where M_i represents the ith column of intermediate matrix and $\widetilde{R}_i$ denotes the ith row of 4x4 residual matrix. When using each row indexed by j ($j = 0...3$) from the transform matrix to multiply $\widetilde{R}_i$, the sum results can also be obtained as follows.

The calculation of even rows with indexes 0, 2 can be summarized as the same group, and the results are

$$m_{0j} = ((\widetilde{r}_{i0} + \widetilde{r}_{i3}) + (\widetilde{r}_{i1} + \widetilde{r}_{i2})) * 64\ , \tag{5.21}$$

$$m_{2j} = ((\widetilde{r}_{i0} + \widetilde{r}_{i3}) - (\widetilde{r}_{i1} + \widetilde{r}_{i2})) * 64\ . \tag{5.22}$$

The calculation of odd rows with indexes 1, 3 can also be summarized as the same group, take 3rd row as an example and the result is

$$m_{3j} = (\widetilde{r}_{i0} - \widetilde{r}_{i3}) * 36 + (\widetilde{r}_{i1} - \widetilde{r}_{i2}) * (-83)\ . \tag{5.23}$$

Based on Eqs. (5.21), (5.22), and (5.23), the calculation of 1-D 4x4 transform can be implemented through the flow-group depicted in Figure 5.3.

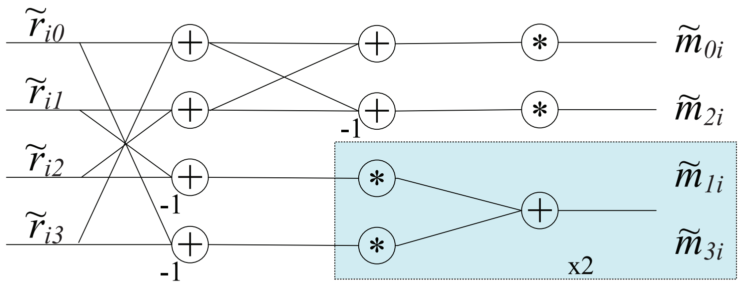

Figure 5.3: Flow-chart of the 1-D 4x4 transform.

Combine Figures 5.3 and 5.1 together, it is easily found out that the 4x4 transform structure is part of the 8x8 transform flow-chart. The combined result is depicted in Figure 5.4.

The proposed unified hardware architecture of the 8x8 and 4x4 transforms is illustrated in Figure 5.5. The "n-point Adder cluster" blocks denote a series of adders to complete certain addition and subtraction operations over the input, and parts of the results from these blocks will be multiplied with the coefficients of the transform matrixes in corresponding "n-point Multi. cluster" blocks. Then, in the "Adder tree n" blocks, the sums of the multiplication

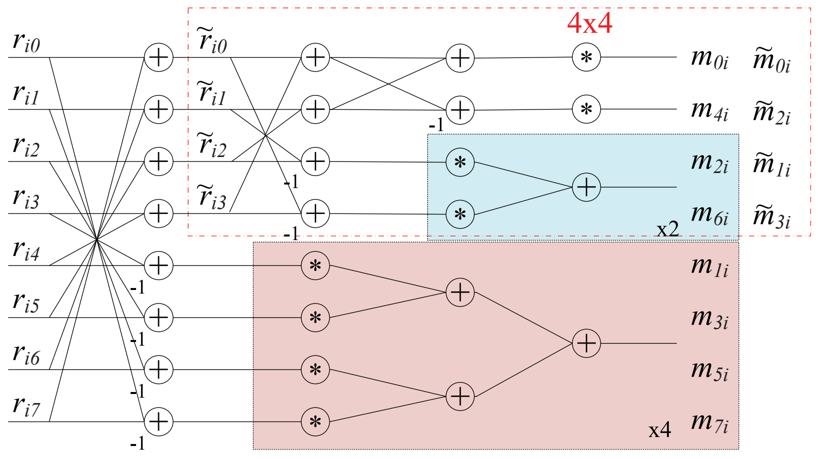

Figure 5.4: Combined flow-chart of the 1-D 8x8 and 4x4 transform.

results will be made. Moreover, we can see that the architecture designed for 4x4 transform is embedded in that of 8x8.

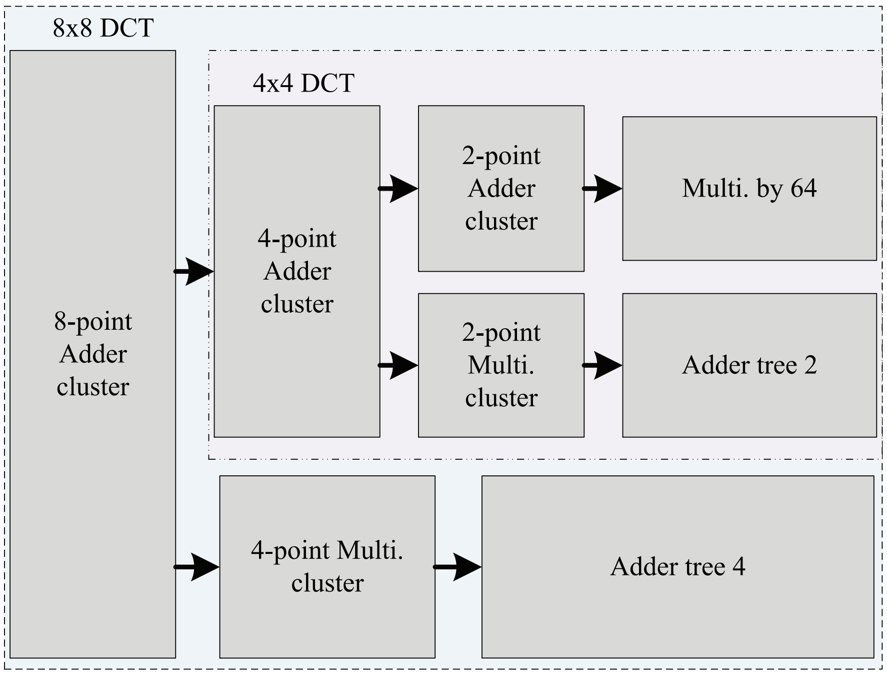

Figure 5.5: Hardware architecture for the 8x8 transform in HEVC.

By the same analysis method, the hardware of the proposed architecture can be reused by transforms of sizes from 4x4 up to 32x32.

5.3.2 Transposition memory design

According to Eq. (5.9), for each cycle, all the pixels from a whole row of the residual block are sent to the transform engine, and the output from it is a whole column of the transformed coefficient block. As depicted in Figure 5.6, the indexes of the pixels denote the order of the input/output to/from the transform engine. According to Eq. (5.10), the second 1-D transform can only start after getting a complete row of the first 1-D transform results, and the results generated from the first 1-D transform should first be stored then used as the input to the second 1-D transform.

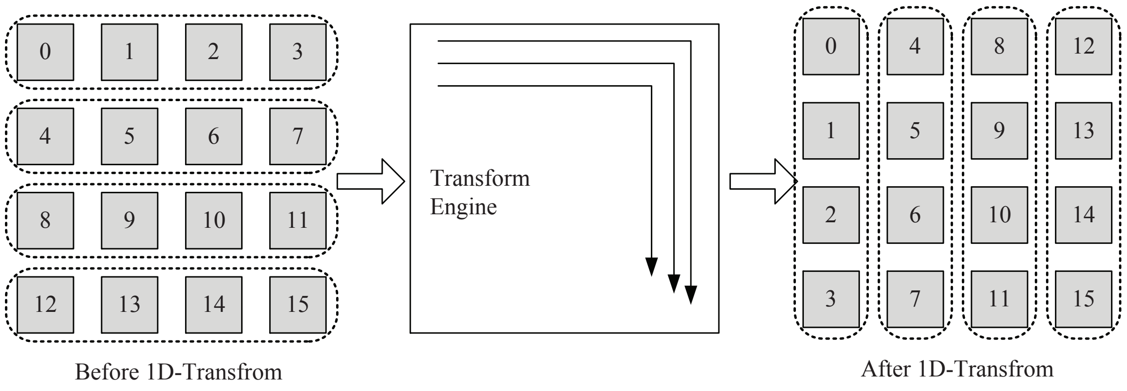

Figure 5.6: The order of the pixels during transform process.

Therefore, a transposition memory is designed following the first 1-D transform operation, and the size of this transposition memory is also 32x32, which can also be reused by 4x4 up to 32x32 transforms as the proposed architecture does. Part of the proposed transposition memory architecture with a size of 4x4 is shown in Figure 5.7.

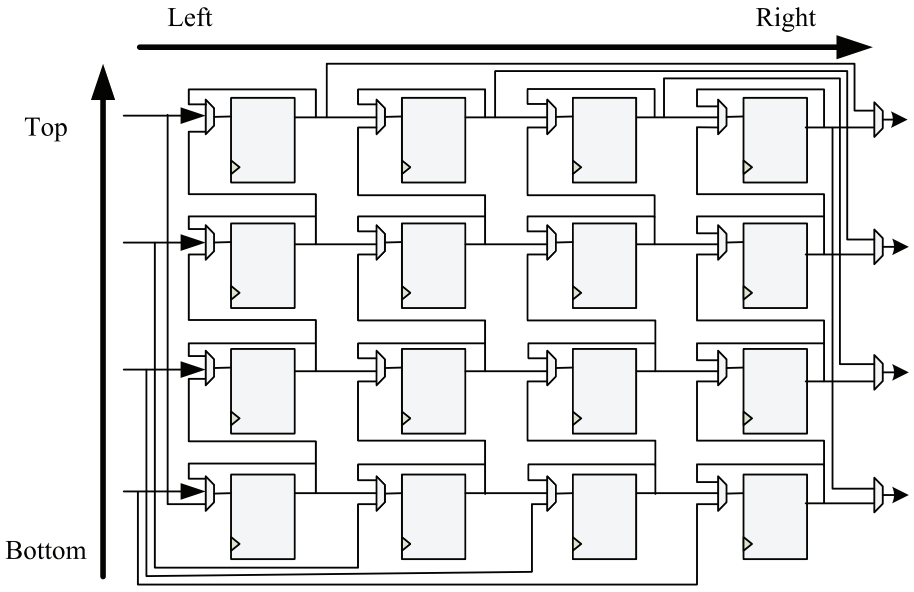

Figure 5.7: 4x4 transposition memory architecture.

The whole transposition memory is composed of 32x32 identical registers, and each transpose register contains a three input multiplexer and a register. The register is a 16-bit width register and 16-bit width is defined by the data width outputted from the first 1-D transform stage as explained in Section 5.2.2.

The multiplexer of each transpose register is designed to control the state and the data flow direction of the transposition memory. When the first input is selected, which is a self feedback from its output, the data inside this memory remains unchanged. While if the second input, which is an output from the register located by its left side, is selected, the data will shift from the left to right. If the third one is chosen, the output from the bottom register will be transmitted to the top one, allowing data to shift from bottom to top.

Finally, the output from this memory is controlled by a two input multiplexer to select the proper one. This designed architecture performs the transpose operation under the same clock as the transform stage used.

The whole architecture for the transform in HEVC is depicted in Figure 5.8. The transposition memory shown in Figure 5.7 is added between the first and second 1-D transforms as analyzed above.

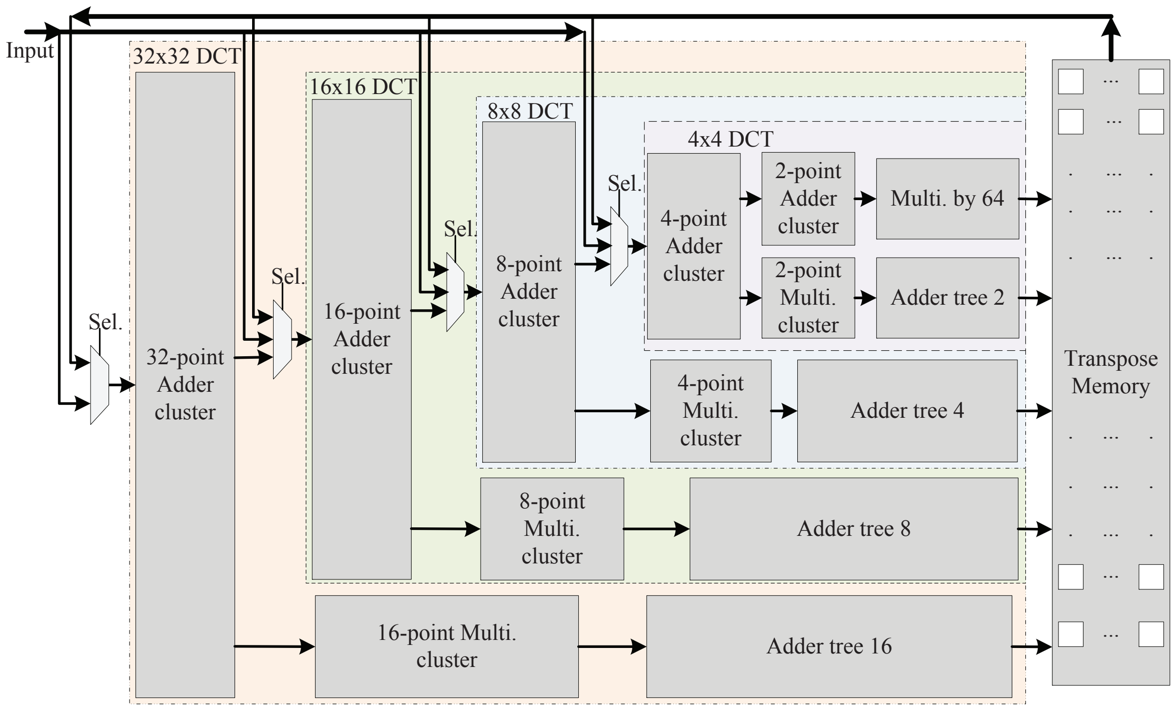

Figure 5.8: Whole architecture for transform in HEVC.

According to Eqs. (5.9) and (5.10), we can find out that the first and second 1-D transforms use the exact same transform matrixes. Therefore, these two 1-D transforms can share the same structure proposed in the above sections. A multiplexer is used to select the proper input between the original residual data and the data from transposition memory, as the “Sel.” signal given in Figure 5.8. Furthermore, this signal can be used to select the transform size. After the second 1-D transform, the resulting transform coefficients are also scaled using right shift operation to 16 bit. This also ensures a bit width of 16 after each transform stage.

5.4 Synthesis results

A detailed complexity analysis is conducted comparing to the original transform design in the HEVC reference software HM [3]. The numbers of different arithmetic operations for 2-D forward transforms of various sizes are shown in Table 5.3. As a fact that multiplication is converted into shift and sum operations, the numbers of these are increased while the number of multiplication is zero.

Table 5.3: Number of arithmetic operations used to perform the transform.

Transform size	HM [3]			Proposed		
	Multiplication	Addition	Shift	Multiplication	Addition	Shift
4x4	48	64	32	0	128	128
8x8	352	448	128	0	1152	1024
16x16	2752	3200	512	0	9216	7872
32x32	21888	23808	2048	0	68608	60928

The functional correctness of the proposed architecture is first simulated using Altera Modelsim software. The test vectors used by the testbench are directly dumped from the HEVC reference software HM, and the output from this proposed transform architecture is compared with the calculation results generated by the same reference software HM.

The proposed architecture is synthesized with Altera Quartus II software, and the selected FPGA is from Cyclone IV E serials with device number: EP4CE115F29I8L. The synthesis result is listed in Table 5.4. After synthesis, timing simulation with a 125 MHz clock using netlist and structure delay information is conducted, and the results are also checked using the reference software HM.

Table 5.4: Synthesis results on FPGA.

Condition	Frequency (MHz)	Logic Elements
Slow 1000mV 100C Model	128.73	40541
Slow 1000mV -40C Model	134.05	
Timing Simulation	125	-

The results show that the designed architecture achieves a maximum operation frequency of 134 MHz.

In Table 5.5, the estimated throughput of the proposed architecture is listed.

The HEVC standard is targeting at videos with higher resolution. Take Class A sequences defined in HM common test conditions [123] as an example, that is 2560x1600 pixels, 30 frames per second with a 4:2:0 sampling format, a throughput of 184.32 Msamlpes/s is required. According to Table 5.5, the proposed design can process high definition sequences in real time.

Table 5.5: Estimated throughput (Msamlpes/s) at 125 MHz on FPGA.

Transform size	Cycle	Throughput	Weight[*]	Average
32x32	75	1707.91	1/85	238.13
16x16	41	780.49	4/85	
8x8	23	347.83	16/85	
4x4	13	153.85	64/85	

[*] weight is determined by the size of each block

Table 5.6 gives a performance comparison between [92] and the proposed design. In [92], a 16 point 1-D transform architecture is designed, so the number of Logic elements in our design is scaled, based on the add operations relationship given in Table 5.3, to keep a reasonable comparison. Results indicate that the proposed design can double (780.49/376.2) the throughput with almost same (about 101.9%)hardware cost.

Table 5.6: Performance comparison with previous work on FPGA.

Design	Jeske [92]	Proposed
FPGA	Cyclone II	Cyclone IV
Structure	Combination	Pipeline
Transform Type	1-D Transform	2-D Transform
Logic elements	5343	5446[*]
Frequency (MHz)	23.51	125
Throughput	376.2	780.49

[*] $5446 \doteq 40541 * (9216/68608)$

The proposed architecture is then synthesized using 45nm technology, and efficiency comparison results are listed in Table 5.7. As it is stated in [95], one SRAM bit is equivalent to 10 2-input NAND gates in terms of silicon area. Concerning the hardware efficiency of a design, we adopt a performance metric called data throughput per unit area (DTUA), which is defined as the ratio of data throughput over the hardware cost given in terms of gate count. When DTUA is adopted as the comparison metric, the higher the value of DTUA is, the more efficient the design is. As a fact that the transform coefficients used for inverse transform are the same as those for forward. So it will still make sense to compare the hardware efficiency, and the proposed architecture is more than 5 (155.6/29.7) times more efficient than the previous design.

The efficiency of the proposed transpose memory is also analyzed and results in Table 5.8 indicate that this design can fulfill the transpose operation in less clock cycles while remaining a fewer hardware cost.

In Table 5.9 the estimated throughput of the proposed architecture is calculated. Results show that it can support real-time process of 4Kx2K (4096x2048, 30fps, 4:2:0 format) video sequences.

Table 5.7: Efficiency comparison with previous work on ASIC platform.

Design	Shen [95]	Proposed
Gate Count	109.2K	205.5K
SRAM (bit)	2560	-
Total Area (gate)	134.8K	205.5K
Frequency (MHz)	350	333
Throughput (pixels/cycle)	4	32
Latency (cycle)	256+5	32+6
DTUA (pixels/cycle/gate)	29.7×10^{-6}	155.6×10^{-6}

Table 5.8: Transpose memory comparison with previous work on ASIC platform.

Design	Shen [95]	Proposed
Gates	-	139.5K
SRAM (bit)	16384	-
Total Area	163.8K	139.5K
Frequency (MHz)	-	333
Latency	256	32

Table 5.9: Estimated throughput (Msamlpes/s) at 333MHz on ASIC platform.

Transform size	Cycle	Throughput	Weight	Average
32x32	75	4546.56	1/85	634.35
16x16	41	2079.22	4/85	
8x8	23	926.609	16/85	
4x4	13	409.846	64/85	

5.5 Conclusion

This chapter presented a high-performance VLSI architecture of the transform applied in the video coding standard-HEVC. The proposed architecture could support 1-D and 2-D transforms with a variety of transform sizes from 4x4 to 32x32. Synthesis results on FPGA platform indicated that the proposed design could double the throughput, compared to previous work, with almost same hardware cost. Moreover, synthesis results under 45nm technology showed that it could support real-time process of 4Kx2K (4096x2048, 30fps) video sequences. When a comparison index DTUA was adopted, the proposed architecture was almost 5 times more efficient than previous design.

Chapter 6

Conclusion

In this dissertation, a series of fast mode decision algorithms and its corresponding hardware architectures as well as the hardware design of the transform of H.265/HEVC were proposed.

In Chapter 1, background information and previous researches as well as the objective of this thesis were listed. HEVC had incorporated a series of the state-of-the-art technologies and algorithms which increased the computational complexity. To find the best encoding mode for a certain block, a huge number of combinations of block sizes and candidate modes had to be checked, which was very time-consuming. Hence, in this work, a number of fast mode decision algorithms were proposed to accelerate the mode decision process.

In Chapter 2, a brief introduction about the H.265/HEVC was presented. During the prediction and transform coding processes, a flexible quadtree based block partition scheme was employed. Approaches, namely Tiles and WPP, were taken in to enhance the parallel scalability. For the prediction coding, an enhanced intra prediction with 35 candidate modes as well as a flexible motion-compensation prediction with merging technology were employed. To encode the residual, larger sized transform and advanced CABAC engine were applied, and a de-blocking filter and a newly SAO filter were used to filter the reconstructed samples.

In Chapter 3, a course of hierarchical structure based low complexity fast mode decision algorithms were presented. The co-located depth information from previous frame was used to predict the split structure of current LCU. The residual generated by inter prediction was analyzed to determine the encoding flow. A hardware-oriented low complexity fast intra prediction algorithm was proposed. This proposed algorithm adopted a fast DCD to detect the dominant direction of the CU. Moreover, four simple but efficient early termination strategies were proposed to terminate the RDO process properly. Simulation results showed that the proposed overall algorithm reduced the encoding time by 54.0 ~ 68.4%, without introducing any noticeable performance degradation.

In Chapter 4, the hardware architectures of the fast mode decision algorithms were discussed. A state machine based module for the depth prediction combined with the residual check algorithm was presented, and the corresponding system framework as well as the realization of the internal architecture of this module were analyzed in detail. Hardware synthesis results demonstrated that the proposed architecture achieved a maximum frequency of about 193 MHz. The hardware implementation of the proposed fast DCD algorithm along with two previous works was presented. The performance comparison results showed that this proposed algorithm achieved almost the same time reduction while maintained a lower bitrate loss compared with other work. Furthermore, our low complexity algorithm could save more than 88.9% resources when implemented on the hardware platform.

Chapter 5 presented a high-performance VLSI architecture of the transform applied in HEVC . The proposed architecture could support 1-D and 2-D transforms with a variety of transform sizes from 4x4 to 32x32. Synthesis results on FPGA platform indicated that the proposed design could double the throughput, compared to the previous work, with almost same hardware cost. Moreover, synthesis results under 45nm technology showed that it could support real-time process of 4Kx2K (4096x2048, 30fps) video sequences. When a comparison index DTUA was adopted, the proposed architecture was almost 5 times more efficient than the previous design.

One of the future works of this thesis is to design the remaining parts of the proposed hardware system, including buffer systems, and the coding engines, as shown in Figure 4.1. For the hardware implementation, the following aspects should be considered carefully:

(1) The memory cost introduced by storing the maximum depth information of the co-located LCU from the previous frame.
(2) The tradeoff between the cost resulting from requiring data from previous frame and the complexity reduction.
(3) The buffer system used to compensate the fluctuation of the processing time for each frame due to the periodic on/off of depth prediction.
(4) The allocation of hardware resources for the frames encoded with or without depth prediction.

As discussed in Chapter 5, the hardware architecture of the forward transform had been proposed. As another future work, the hardware design of the inverse transform is an interesting and important topic to study. As the transform matrixes used by both forward and inverse transforms are identical, the design of a unique architecture that supports both forward and inverse transforms with all possible sizes is also a challenging research topic.

Bibliography

[1] "Advanced video coding for generic audio-visual services," *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC)*, Mar. 2005.

[2] "High efficiency video coding," *ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC)*, Jan. 2013.

[3] JCT-VC, "High efficiency video coding (HEVC) test model 10 (HM 10) encoder description," *JCTVC-L1002*, Jan. 2013.

[4] J. Vanne, M. Viitanen, and T. Hamalainen, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885 - 1898, Dec. 2012.

[5] F. Bossen, B. Bross, and K. Suhring, "HEVC Complexity and Implementation Analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685 - 1696, Dec. 2012.

[6] J. Ohm, G. Sullivan, and H. Schwarz, "Comparison of the coding efficiency of video coding standards - including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669 - 1684, Dec. 2012.

[7] G. Correa, P. Assuncao, and L. Agostini, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1899 - 1909, Dec. 2012.

[8] G. Sullivan, and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74 - 90, Nov. 1998.

[9] A. Ortega, and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23 - 50, Nov. 1998.

[10] C. Duanmu, and X. Chen, "Fast motion estimation mode decision algorithm for H.264/AVC video coding standard," In *Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PrimeAsia)*, pp. 313 - 316, Jan. 2009.

[11] K. Chang, B. Yang, and W. Zhang, "Novel fast mode decision algorithm for p-slices in H.264/AVC," In *International Conference on Information Assurance and Security (ISA)*, pp. 189 - 193, Aug. 2009.

[12] L. Salgado, and M. Nieto, "Sequence independent very fast mode decision algorithm on H.264/AVC baseline profile," In *IEEE International Conference on Image Processing (ICIP)*, pp. 41 - 44, Oct. 2006.

[13] Z. Pan, and S. Kwong, "A fast inter-mode decision scheme based on luminance difference for H.264/AVC," In *International Conference on System Science and Engineering (ICSSE)*, pp. 260 - 263, Jun. 2011.

[14] M. Hwang, J. Cho, and J. Kim, "Fast intra prediction mode selection scheme using temporal correlation in H.264," In *IEEE Region 10 Conference (TENCON)*, pp. 1 - 5, Nov. 2005.

[15] C. Lien, C. Chen, and Y. Chang, “Fast intra/inter mode decision for H.264/AVC using the spatial-temporal prediction scheme,” In *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC)*, pp. 194 - 199, Jul. 2009.

[16] B. Hilmi, K. Goswami, and J. Lee, “Fast inter-mode decision algorithm for H.264/AVC using macroblock correlation and motion complexity analysis,” In *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 90 - 91, Jan. 2012.

[17] D. Wu, F. Pan, and K. Lim, “Fast intermode decision in H.264/AVC video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 6, pp. 953 - 958, Jul. 2005.

[18] K. Bharanitharan, B. Liu, and J. Yang, “A low complexity fast inter prediction algorithm for H.264/AVC,” In *International Conference on Computer and Electrical Engineering (ICCEE)*, pp. 73 - 77, Dec. 2008.

[19] A. Yu, and G. Martin, “Advanced block size selection algorithm for inter frame coding in H.264/MPEG-4 AVC,” In *IEEE International Conference on Image Processing (ICIP)*, pp. 95 - 98, Oct. 2004.

[20] H. Wang, J. Lin, and J. Yang, “Fast H.264 inter mode decision based on inter and intra block conditions,” In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 3647 - 3650, May 2007.

[21] G. Bao, and H. Hu, “An optimized method of inter modes selection in H.264/AVC,” In *International Symposium on Intelligence Information Processing and Trusted Computing (IPTC)*, pp. 611 - 614, Oct. 2010.

[22] I. Choi, J. Lee, and B. Jeon, “Fast coding mode selection with rate-distortion optimization for MPEG-4 Part-10 AVC/H.264,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 12, pp. 1557 - 1561, Dec. 2006.

[23] T. Zhao, H. Wang, and S. Kwong, “Fast mode decision based on mode adaptation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 5, pp. 697 - 705, May 2010.

[24] P. Lee, H. Chang, and S. Huang, “Coding mode determination by using fuzzy logic in H.264 motion estimation,” In *Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, pp. 1 - 6, Jun. 2009.

[25] A. Yu, “Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC,” In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 169 - 172, May 2004.

[26] B. Zhan, B. Hou, and R. Sotudeh, “Statistical data based low-complexity mode decision for P-frame encoding in H.264/AVC,” In *International Conference on Information, Communications & Signal Processing (ICICS)*, pp. 1 - 5, Dec. 2007.

[27] F. Pan, X. Lin, and S. Rahardja, “Fast mode decision algorithm for intra prediction in H.264/AVC video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 813 - 822, Jul. 2005.

[28] C. Hsia, J. Chiang, and Y. Wang, “Fast intra prediction mode decision algorithm for H.264/AVC video coding standard,” In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP)*, pp. 535 - 538, Nov. 2007.

[29] C. Miao and C. Fan, “Extensive pixel-based fast direction detection algorithm for H.264/AVC intra mode decision,” In *IEEE Region 10 Conference (TENCON)*, pp. 1636 - 1640, Nov. 2010.

[30] K. Bharanitharan, and A. Tsai, “Efficient block size decision algorithm for intra mode

decision in H.264/AVC encoder," In *IEEE International Symposium on Multimedia (ISM)*, pp. 96 - 99, Dec. 2009.

[31] J. Xin, and A. Vetro, "Fast mode decision for intra-only H.264/AVC coding," In *Picture Coding Symposium (PCS)*, pp. 34 - 39, Apr. 2006.

[32] M. Jafari, and S. Kasaei, "Fast intra-prediction mode decision in H.264 advanced video coding," In *IEEE Singapore International Conference on Communication systems (ICCS)*, pp. 1 - 6, Oct. 2006.

[33] Y. Ding, Y. Si, and C. Yao, "Fast intra mode decision algorithm for H.264/AVC," In *Congress on Image and Signal Processing (CISP)*, pp. 570 - 574, May 2008.

[34] H. Zeng, K. Ma, and C. Cai, "Hierarchical intra mode decision for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 6, pp. 907 - 912, Jun. 2010.

[35] J. Li, and L. Chen, "A fast intramode decision algorithm for H.264/AVC," In *IEEE International Conference on Communication Technology (ICCT)*, pp. 416 - 419, Nov. 2010.

[36] Y. Cheng, and M. Wu, "A fast mode decision algorithm of 4x4 block intra prediction for H.264/AVC," In *International Conference on Intelligent System and Knowledge Engineering (ISKE)*, pp. 1231 - 1236, Nov. 2008.

[37] G. Hwang, J. Park, and B. Jung, "Efficient fast Intra mode decision using transform coefficients," In *International Conference on Advanced Communication Technology (ICACT)*, pp. 399 - 402, Feb. 2007.

[38] C. Tseng, H. Wang, and J. Yang, "Enhanced intra-4x4 mode decision for H.264/AVC coders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp. 1027 - 1032, Aug. 2006.

[39] Y. Shi, O. Au, and X. Zhang, "Content based fast prediction unit quadtree depth decision algorithm for HEVC," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 225 - 228, May 2013.

[40] L. Shen, Z. Zhang, and P. An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 207 - 213, Mar. 2013.

[41] H. Lee, K. Kim, and T. Kim, "Fast encoding algorithm based on depth of coding-unit for high efficiency video coding," *Optical Engineering*, vol. 51, no. 6, pp. 1 - 11, Jun. 2012.

[42] L. Shen, Z. Liu, and X. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465 - 470, Feb. 2013.

[43] J. Leng, L. Sun, and T. Ikenaga, "Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC," In *International Conference on Multimedia and Signal Processing (CMSP)*, pp. 56 - 59, May 2011.

[44] S, Tai, C. Chang, and B. Chen, "Speeding up the decisions of quad-tree structures and coding modes for HEVC coding units," In *Proceedings of the International Computer Symposium (ICS)*, pp. 393 - 401, Dec. 2012.

[45] M. Cassa, M. Naccari, and F. Pereira, "Fast rate distortion optimization for the emerging HEVC standard," In *Picture Coding Symposium (PCS)*, pp. 493 - 496, May 2012.

[46] G. Tian, and S. Goto, "Content adaptive prediction unit size decision algorithm for HEVC intra coding," In *Picture Coding Symposium (PCS)*, pp. 405 - 408, May 2012.

[47] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on

Bayesian decision rule," In *Picture Coding Symposium (PCS)*, pp. 453 - 456, May 2012.

[48] J. Kim, J. Yang, and K. Won, "Early determination of mode decision for HEVC," In *Picture Coding Symposium (PCS)*, pp. 449 - 452, May 2012.

[49] JCT-VC, "Early skip detection for HEVC," *JCTVC-G543*, Nov. 2011.

[50] JCT-VC, "Early termination of CU encoding to reduce HEVC complexity," *JCTVC-F045*, Jul. 2011.

[51] JCT-VC, "Coding tree pruning based CU early termination," *JCTVC-F092*, Jul. 2011.

[52] J. Lee, C. Park, and B. Kim, "Fast coding algorithm based on adaptive coding depth range selection for HEVC," In *IEEE International Conference on Consumer Electronics(ICCE)*, pp. 31 - 33, Sept. 2012.

[53] J. Lee, C. Kim, and J. Lee, "Fast coding algorithm for high efficient video coding (HEVC)," In *International Conference on Ubiquitous Information Technologies and Applications (CUTE)*, pp. 289 - 297, 2012.

[54] H. Tan, F. Liu, and Y. Tan, "On fast coding tree block and mode decision for high-efficiency video coding (HEVC)," In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 825 - 828, Mar. 2012.

[55] K. Choi, and E. Jang, "Fast coding unit decision method based on coding tree pruning for high efficiency video coding," *Optical Engineering*, vol. 51, no. 3, pp. 1 - 3, Mar. 2012.

[56] J, Kim, S. Jeong, and S. Cho, "Adaptive coding unit early termination algorithm for HEVC," In *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 261 - 262, Jan. 2012.

[57] JCT-VC, "Encoder speed-up for the motion vector predictor cost estimation," *JCTVC-H0178*, Feb. 2012.

[58] H. Wang, Y. Wee, and Y. Kim, "An early termination method using the residual in high efficiency video coding," In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1 - 2, Jun. 2012.

[59] S. Teng, H. Hang, and Y. Chen, "Fast mode decision algorithm for residual quadtree coding in HEVC," In *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1 - 4, Nov. 2011.

[60] K. Choi, and E. Jang, "Early TU decision method for fast video encoding in high efficiency video coding," *Electronics Letters*, vol. 48, no. 12, pp. 689 - 691, Jun. 2012.

[61] P. Chiang, and T. Chang, "Fast zero block detection and early CU termination for HEVC Video Coding," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1640 - 1643, May 2013.

[62] A. Motra, A. Gupta, and M. Shukla, "Fast intra mode decision for HEVC video encoder," In *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1 - 5, Sep. 2012.

[63] T. Silva, L. Cruz, and L. Agostini, "Fast HEVC intra mode decision based on dominant edge evaluation and tree structure dependencies," In *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 568 - 571, Dec. 2012.

[64] W. Jiang, H. Ma, and Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," In *International Conference on Consumer Electronics Communications and Networks (CECNet)*, pp. 1836 - 1840, Apr. 2012.

[65] Y. Zhang, Z. Li, and B. Li, "Gradient-based fast decision for intra prediction in HEVC,"

In *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1 - 6, Nov. 2012.

[66] G. Chen, Z. Liu, and T. Ikenaga, "Fast HEVC intra mode decision using matching edge detector and kernel density estimation alike histogram generation," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 53 - 56, May 2013.

[67] L. Zhao, L. Zhang, and S. Ma, "Fast mode decision algorithm for intra prediction in HEVC," In *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1 - 4, Nov. 2011.

[68] Y. Kim, D. Jun, and S. Jung, "A fast intra prediction method using hadamard transform in high efficiency video coding," In *Proceedings of Visual Information Processing and Communication III*, pp. 1 - 10, Feb. 2012.

[69] M. Zhang, C. Zhao, and J. Xu, "An adaptive fast intra mode decision in HEVC," In *IEEE International Conference on Image Processing (ICIP)*, pp. 221 - 224, Sep. 2012.

[70] S. Yan, L. Hong, and W. He, "Group-based fast mode decision algorithm for intra prediction in HEVC," In *International Conference on Signal Image Technology and Internet Based Systems (SITIS)*, pp. 225 - 229, Nov. 2012.

[71] J. Kim, J. Yang, and H. Lee, "Fast intra mode decision of HEVC based on hierarchical structure," In *International Conference on Information, Communications and Signal Processing (ICICS)*, pp. 1 - 4, Dec. 2011.

[72] Y. Cheng, G. Teng, and X. Shi, "A fast intra prediction algorithm for HEVC," In *International Forum on Digital TV and Wireless Multimedia Communication (IFTC)*, pp. 292 - 298, Nov. 2012.

[73] W. Zhao, L. Shen, and Z. Cao, "Texture and correlation based fast intra prediction algorithm for HEVC," In *International Forum on Digital TV and Wireless Multimedia Communication (IFTC)*, pp. 284 - 291, Nov. 2012.

[74] H. Zhang, and Z. Ma, "Fast intra prediction for high efficiency video coding," In *Pacific-Rim Conference on Multimedia (PCM)*, pp. 568 - 577, Dec. 2012.

[75] H. Zhao, and Z. Ma, "Early termination schemes for fast intra mode decision in high efficiency video coding," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 45 - 48, May 2013.

[76] H. Sun, D. Zhou, and S. Goto, "A low-complexity HEVC Intra prediction algorithm based on level and mode filtering," In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1085 - 1090, Jul. 2012.

[77] S. Cho, and M. Kim, "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23 , no. 9, pp. 1555 - 1564, Sept. 2013.

[78] C. Zhou, X. Tian, and Y. Chen, "A fast intra coding unit size decision based on statistical learning for HEVC," In *Proceedings of Applied Mechanics and Materials*, pp. 2107 - 2111, Feb. 2013.

[79] J. Xiong, "Fast coding unit selection method for high efficiency video coding intra prediction," *Optical Engineering*, vol. 52, no. 7, pp. 1 - 10, Jul. 2013.

[80] JCT-VC, "Unified transform design for HEVC with 16 bit intermediate data representation," *JCTVC-D224*, Jan. 2011.

[81] JCT-VC, "Transform design for HEVC with 16 bit intermediate data representation," *JCTVC-E243*, Mar. 2011.

[82] JCT-VC, "CE10: Core transform design for HEVC," *JCTVC-F446*, Jul. 2011.

[83] JCT-VC, "CE10: Core transform design for HEVC," *JCTVC-G495*, Nov. 2011.

[84] C. Fan, F. Li, and G. Shi, “A low complexity multiplierless transform coding for HEVC,” In *Pacific-Rim Conference on Multimedia (PCM)*, pp. 578 - 586, Dec. 2012.

[85] Z. Cheng, C. Chen, and B. Liu, “High throughput 2-D transform architectures for H.264 advanced video coders,”In *IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 1141 - 1144, Dec. 2004.

[86] H. Lin, Y. Chao, and C. Chen, “Combined 2-D transform and quantization architectures for H.264 video coders,” In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1802 - 1805, May 2005.

[87] R. Kordasiewicz, and S. Shirani, “Hardware implementation of the optimized transform and quantization blocks of H.264,” In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 943 - 946, May 2004.

[88] T. Wang, Y. Huang, and H. Fang, “Parallel 4x4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264,” In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. II-800 - II-803, May 2003.

[89] L. Liu, L. Qiu, and M. Rong, “A 2-D forward/inverse integer transform processor of H.264 based on highly-parallel architecture,” In *IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC)* , pp. 158 - 161, Jul. 2004.

[90] K. Chen, J. Guo, and J. Wang, “A high-performance direct 2-D transform coding IP design for MPEG-4 AVC/H.264,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16 , no. 4, pp. 472 - 483, Apr. 2006.

[91] P. Meher, S. Park, and B. Mohanty, “Efficient integer DCT architectures for HEVC,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24 , no. 1, pp. 168 - 178, Jan. 2014

[92] R. Jeske, J. de Souza, and G. Wrege, “Low cost and high throughput multiplierless design of a 16 point 1-D DCT of the new HEVC video coding standard,” In *IEEE Southern Conference on Programmable Logic (SPL)*, pp. 1 - 6, Mar. 2012.

[93] J. Zhu, Z. Liu, and D. Wang, “Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC Codec,” In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 677 - 680, May 2013.

[94] S. Park, and P. Meher, “Flexible integer DCT architectures for HEVC,” In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1376 - 1379, May 2013.

[95] S. Shen, W. Shen, and Y. Fan, “A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards,” In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 788 - 793, Jul. 2012.

[96] S. Shen, W. Shen, and Y. Fan, “A unified forward/inverse transform architecture for multi-standard video codec design,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A, no. 7, pp. 1534 - 1542, Jul. 2013.

[97] M. Martuza, and K. Wahid, “A cost effective implementation of 8x8 transform of HEVC from H.264/AVC,” In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1 - 4, May 2012.

[98] “Generic coding of moving pictures and associated audio information–Part 2: Video,” *ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG 2 Video)*, Nov. 1994.

[99] “Coding of audio-visual objects–Part 2: Visual,” *ISO/IEC 14496-2 (MPEG-4 Visual)*, Apr. 1999.

[100] “The information technology advanced audio and video coding–Part 2: Video,” *GB/T*

200090.2-2006 (AVS-P2), Mar. 2006.

[101] "VC-1 Compressed video bitstream format and decoding process," *SMPTE 421M*, Apr. 2006.

[102] M. Zhou, W. Gao, and M. Jiang, "HEVC lossless coding and improvements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1839 - 1843, Dec. 2012.

[103] G. Sullivan, J. Ohm, and W. Han, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649 - 1668, Dec. 2012.

[104] M. Pourazad, C. Doutre, and M. Azimi, "HEVC: the new gold standard for video compression," *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36 - 46, Jul. 2012.

[105] Y. Yuan, I. Kim, and X. Zheng, "Quadtree based nonsquare block structure for inter frame coding in high efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1707 - 1719, Dec. 2012.

[106] I. Kim, J. Min, and T. Lee, "Block partitioning structure in the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1697 - 1706, Dec. 2012.

[107] C. Chi, M. Alvarez-Mesa, and B. Juurlink, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1827 - 1838, Dec. 2012.

[108] J. Lainema, F. Bossen, and W. Han, "Intra coding of the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792 - 1801, Dec. 2012.

[109] JCT-VC, "Enhancements to intra coding," *JCTVC-D235*, Jan. 2011.

[110] P. Helle, S. Oudin, and B. Bross, "Block merging for quadtree-based partitioning in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1720 - 1731, Dec. 2012.

[111] V. Sze, and M. Budagavi, "High throughput CABAC entropy coding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1778 - 1791, Dec. 2012.

[112] J. Sole, R. Joshi, and N. Nguyen, "Transform coefficient coding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1765 - 1777, Dec. 2012.

[113] C. Yeo, Y. Tan, and Z. Li, "Dynamic range analysis in high efficiency video coding residual coding and reconstruction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1131 - 1136, Jul. 2012.

[114] A. Norkin, G. Bjontegaard, and A. Fuldseth, "HEVC deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1746 - 1754, Dec. 2012.

[115] C. Fu, E. Alshina, and A. Alshin, "Sample adaptive offset in the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755 - 1764, Dec. 2012.

[116] R. Sjoberg, Y. Chen, and A. Fujibayashi, "Overview of HEVC high-level syntax and reference picture management," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1858 - 1870, Dec. 2012.

[117] H. Li, B. Li, and J. Xu, "Rate-distortion optimized reference picture management for high efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1844 - 1857, Dec. 2012.

[118] W. Zhao, T. Onoye, and T. Song, "Hierarchical structure based fast mode decision for H.265/HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, (in press).

[119] W. Zhao, T. Onoye, and T. Song, "Hardware architecture of the fast mode decision algorithm for H.265/HEVC," In *IEEE International Conference on Image Processing (ICIP)*, Oct. 2014.

[120] W. Zhao, T. Onoye, and T. Song, "Hardware-oriented fast mode decision algorithm for intra prediction in HEVC," In *Picture Coding Symposium (PCS)*, pp. 109 - 112, Dec. 2013.

[121] W. Zhao, T. Onoye, and T. Song, "High-performance multiplierless transform architecture for HEVC," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1668 - 1671, May 2013.

[122] W. Zhao, and T. Onoye, "A high-performance multiplierless hardware architecture of the transform applied to H.265/HEVC emerging video coding standard," *IEICE Technical Report*, vol. 112, no. 207, SIS2012-18, pp. 11 - 16, Sep. 2012.

[123] JCT-VC, "Common HM test conditions and software reference configurations," *JCTVC-K1100*, Oct. 2012.

[124] G. Bjontegaard, "Calculation of average PSNR differences between RDcurves," *ITU-T SG16 Q.6, VCEG-M33*, Apr. 2001.