ALGORITHMS AND HARDWARE CO-DESIGN OF HEVC INTRA ENCODERS

by

Yuanzhi Zhang

B.S., Shandong University, 2011
M.S., Shandong University, 2014

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Doctor of Philosophy Degree

Department of Electrical and Computer Engineering
in the Graduate School
Southern Illinois University Carbondale
December 2019

DISSERTATION APPROVAL

ALGORITHMS AND HARDWARE CO-DESIGN OF HEVC INTRA ENCODERS

by

Yuanzhi Zhang

A Dissertation Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in the field of Electrical and Computer Engineering

Approved by:

Dr. Chao Lu, Chair

Dr. Spyros Tragoudas

Dr. Haibo Wang

Dr. Lalit Gupta

Dr. Mingqing Xiao

Graduate School
Southern Illinois University Carbondale
May 13, 2019

AN ABSTRACT OF THE DISSERTATION OF

Yuanzhi Zhang, for the Doctor of Philosophy degree in Electrical and Computer Engineering, presented on May 13, 2019, at Southern Illinois University Carbondale.

TITLE: ALGORITHMS AND HARDWARE CO-DESIGN OF HEVC INTRA ENCODERS

MAJOR PROFESSOR: Dr. Chao Lu

Digital video is becoming extremely important nowadays and its importance has greatly increased in the last two decades. Due to the rapid development of information and communication technologies, the demand for Ultra-High Definition (UHD) video applications is becoming stronger. However, the most prevalent video compression standard H.264/AVC released in 2003 is inefficient when it comes to UHD videos. The increasing desire for superior compression efficiency to H.264/AVC leads to the standardization of High Efficiency Video Coding (HEVC). Compared with the H.264/AVC standard, HEVC offers a double compression ratio at the same level of video quality or substantial improvement of video quality at the same video bitrate. Yet, HE-VC/H.265 possesses superior compression efficiency, its complexity is several times more than H.264/AVC, impeding its high throughput implementation. Currently, most of the researchers have focused merely on algorithm level adaptations of HEVC/H.265 standard to reduce computational intensity without considering the hardware feasibility. What's more, the exploration of efficient hardware architecture design is not exhaustive. Only a few research works have been conducted to explore efficient hardware architectures of HEVC/H.265 standard. In this dissertation, we investigate efficient algorithm adaptations and hardware architecture design of HEVC intra encoders. We also explore the deep learning approach in mode prediction.

From the algorithm point of view, we propose three efficient hardware-oriented algorithm adaptations, including mode reduction, fast coding unit (CU) cost estimation, and group-based

CABAC (context-adaptive binary arithmetic coding) rate estimation. Mode reduction aims to reduce mode candidates of each prediction unit (PU) in the rate-distortion optimization (RDO) process, which is both computation-intensive and time-consuming. Fast CU cost estimation is applied to reduce the complexity in rate-distortion (RD) calculation of each CU. Group-based CABAC rate estimation is proposed to parallelize syntax elements processing to greatly improve rate estimation throughput.

From the hardware design perspective, a fully parallel hardware architecture of HEVC intra encoder is developed to sustain UHD video compression at 4K@30fps. The fully parallel architecture introduces four prediction engines (PE) and each PE performs the full cycle of mode prediction, transform, quantization, inverse quantization, inverse transform, reconstruction, rate-distortion estimation independently. PU blocks with different PU sizes will be processed by the different prediction engines (PE) simultaneously. Also, an efficient hardware implementation of a group-based CABAC rate estimator is incorporated into the proposed HEVC intra encoder for accurate and high-throughput rate estimation.

To take advantage of the deep learning approach, we also propose a fully connected layer based neural network (FCLNN) mode preselection scheme to reduce the number of RDO modes of luma prediction blocks. All angular prediction modes are classified into 7 prediction groups. Each group contains 3-5 prediction modes that exhibit a similar prediction angle. A rough angle detection algorithm is designed to determine the prediction direction of the current block, then a small scale FCLNN is exploited to refine the mode prediction.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Chao Lu for his consistent support, encouragement, and guidance throughout my Ph.D. studies. He is a passionate scholar with great patience and immense knowledge. Without his persistent help, this dissertation would not have been possible.

I would like to take an opportunity to honorably thank Dr. Spyros Tragoudas, Dr. Haibo Wang, Dr. Lalit Gupta, and Dr. Mingqing Xiao for taking the time to serve on my committee and their continuous guidance in my dissertation.

I would also like to thank all the members of VLSI Lab (E225) for their kind help and support in my life. Special thanks to my friends, Zichang Zhang and Yan Wu for their help and support.

At last, I would like to thank my family for their unconditional support in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Video compression is playing a more essential role in our daily life due to the increasing importance of video data in our society. The past decade has seen the huge progress achieved in the fields of wireless communication, semiconductor manufacturing, software development, and data science, which empower the realization of compute-intensive and cutting-edge technologies, such as high definition video applications.

Recently, video-based applications have attracted more attention than ever. For example, the online video streaming service has achieved exponential growth when compared with traditional television broadcasting. People are more likely to spend their time with digital media because of its flexible and custom-oriented content service. Emerging applications, such as unmanned aerial vehicle (UAV) [1], autonomous driving [2], motion detection and recognition [3], etc., have been developed based on video data. For example, Video Assistant Referee (VAR) system which is well-known and widely used in live sports, like football, has been developed to assist referees to make correct decisions. All those new applications operate based on real-time video transmission and encoding techniques. The emerging of numerous video data-based applications further leads to the explosive growth of video data. Moreover, the increasing activities of video transmission through network channels (Internet, 3G/4G/5G, etc.) will definitely increase the network burden and result in inevitable network congestion.

Obviously, transmitting or storing raw video data is far away from cost-effective, because redundant video data has not been eliminated at all. In the digital video field, the process of video data redundancy removal is called video compression, also known as video coding.

Currently, the most prevalent video coding standard is H.264 (AVC), which was released in 2003 [4]. There is no doubt that H.264 can greatly reduce the video size, however, when it comes to high definition (HD) videos, H.264 is not very efficient. Because it was designed under the circumstance where video applications were not widespread and the demand for high definition videos was not as strong as today. However, in the past decade, displaying technology has achieved huge progress. Liquid-Crystal Display (LCD) has been widely used to replace cathode-ray-tube (CRT) TVs [5]. Various kinds of Light-emitting Diode (LED) displaying technologies have been invented and adopted by displaying market, like Organic LED and Active-matrix OLED [6, 7, 8]. The development of display technology further increases people's demand for high definition video services that provide extremely high-quality vision experience but also require more advanced compression tools than H.264.

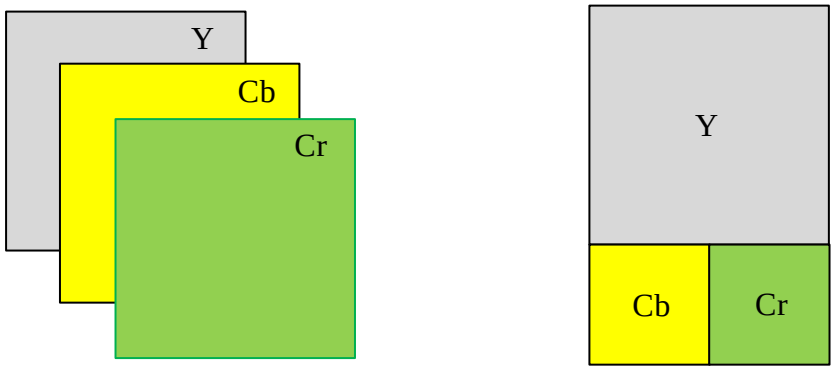
Therefore, High Efficiency Video Coding (HEVC/H.265) standard was proposed and the first version was released in 2013 [9]. HEVC has been announced to provide 50% more bitrate saving at the same level of video quality, compared with H.264 [10]. Despite the advantages that HEVC provides in compression efficiency, its complexity in implementation is much higher than H.264. From an algorithm point of view, a lot of researches have already been conducted to reduce the computational complexity of HEVC standard. While most tried to develop fast algorithms, which simplify mode prediction and motion search, only limited research aimed to find algorithms not only less compute-intensive but also hardware friendly. From the hardware implementation point of view, it is challenging to design an efficient and high-throughput hardware architecture of HE-VC encoder, which is capable of dealing with UHD video compression in real time. Few efforts have been made in academics towards hardware implementation. Therefore, the exploration of efficient HEVC algorithm adaptations and

hardware architecture design is not exhaustive. In this dissertation, we focus on both efficient algorithm adaptations and hardware architecture design of HEVC intra-frame coding, which is the most essential part of the HEVC standard.

In this section, we have briefly introduced the background of video compression. Section 1.2 shows some basic concepts of digital videos and video compression. Then, the challenges of implementing HEVC intra encoder will be introduced in section 1.3. The dissertation outline will be presented in section 1.4.

1.2 FUNDAMENTALS OF VIDEO COMPRESSION

The main purpose of video compression is to minimize transmission bit rate or video file size, meanwhile retaining as high video fidelity as possible. Compression is performed exploiting existing video data redundancy. Two kinds of redundancy are most commonly utilized in video compression: spatial redundancy and temporal redundancy. Spatial redundancy refers to repeated information existing within a frame, while temporal redundancy exists between similar blocks of consecutive frames.



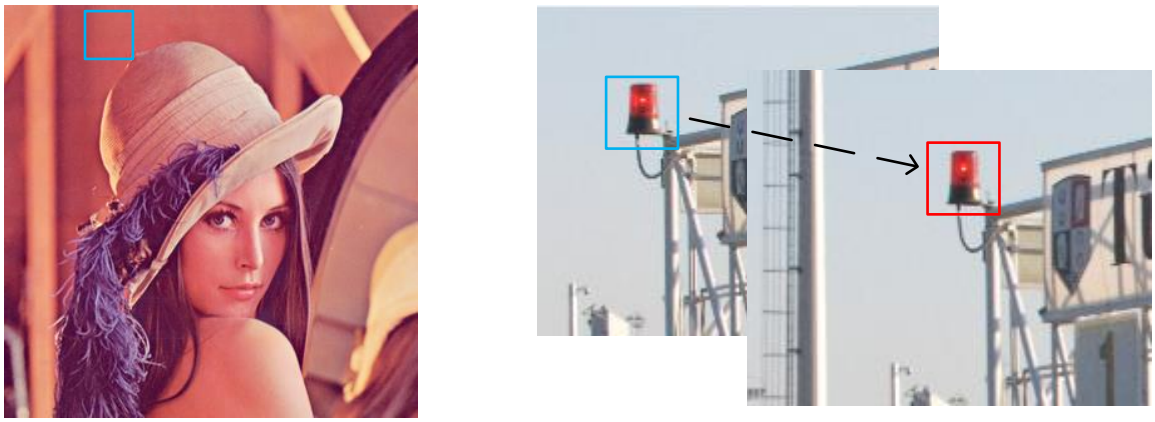
(a) YCbCr 4:4:4 (b) YCbCr 4:2:0

Fig. 1.1. Difference between two raw video formats: YCbCr 4:4:4 and YCbCr 4:2:0.

A frame of digital video typically consists of three rectangular pixel blocks and each pixel is usually an 8-bit integer number representing color value. Each rectangular pixel block is called component, which is one part of a color representation system having three components

called Y, Cb, and Cr [11]. While component Y (also known as luma) represents brightness, the other two components Cb and Cr (known as chroma) represent chrominance. In video compression, format YCbCr is preferred rather than RGB format, because the human visual system is more sensitive to brightness (luma) than chrominance (chroma). Therefore, down sampling usually be applied to chroma blocks to reduce computation with negligible perceptual quality decrease. In this study, each chroma component has quarter pixels of the luma component. This is known as 4:2:0 sub-sampling, which has been widely used as the raw video format in compression. As shown in Fig. 1.1, the total number of pixels in 4:2:0 format is only half of 4:4:4 format. Due to the greatly reduced pixels, 4:2:0 format can drastically save time and computations.

As mentioned above, video compression relies on identifying spatial redundancy and temporal redundancy. Spatial redundancy is defined as redundancy which exists within a single frame. For example, spatial redundancy is always present in areas of video frame where pixel values vary by small amounts, like repeated pixels in smooth background areas as shown in Fig. 1.2(a). Temporal redundancy is defined as redundancy existing between consecutive frames. As shown in Fig. 1.2(b), two image blocks of successive frames show significant similarity, which indicates it is inefficient to compress a similar block twice. According to redundancy utilized in the coding of video frames, video frames can be classified as two categories: intra frame and inter frame. Intra frame refers to frames where only spatial redundancy is utilized during video compression, while both spatial redundancy and temporal redundancy will be exploited in inter frames.



Frame 0 Frame 1

(a) Spatial redundancy (b) Temporal redundancy

Fig. 1.2. Example of spatial and temporal redundancy.

1.3 RESEARCH CHALLENGES

Though HEVC standard provides superior compression efficiency to H.264, it is also a non-negligible fact that HEVC is several times more complex than H.264. HEVC improves its coding efficiency through both introducing new coding algorithms and extending existing coding features.

The basic coding structure of HEVC is still a block-based hybrid video coding approach, which is the same with all major video coding standards [12]. Each coding block is either intra predicted or inter predicted according to the frame type. Signal errors after prediction are further processed through transform coding, where energy is concentrated and less important coefficients can be removed in quantization. Then, entropy encoder processes quantized coefficients together with prediction information to generate output bitstream. Although HEVC has a similar coding procedure as H.264, there are many differences in various aspects.

Firstly, HEVC introduces a new coding structure, where macroblock (MB) is replaced by a coding tree unit (CTU) as the root coding unit. The size of CTU can be configured from up to 64x64 down to 16x16 to cope with different application scenarios. Usually, large CTU size will

lead to better compression performance for high definition videos. Coding unit (CU) is the basic coding element in HEVC and the largest CU size can be the same with CTU size. Then, HEVC introduces a quad-tree structure, where each CU can be recursively split into four sub-CUs until the smallest allowed CU size is met. CU size can be chosen from 64x64 down to 8x8. And each CU contains a prediction unit (PU) and a transform unit (TU). While PU contains both prediction information and prediction block (PB), TU corresponds to transform coding and transform block. The supported PU and TU sizes are from 64x64 to 4x4 and from 32x32 to 4x4, respectively. Moreover, for each luma PB, HEVC extends intra prediction mode from 9 in H.264 to 35. And for chroma PB, 5 intra prediction modes are supported, while only 4 in H.264. In addition to DCT transform, HEVC also supports discrete sine transform (DST), which is exclusively used for luma 4x4 PB. All those enhancements result in a much more complex RDO (rate-distortion optimization) process, which aims to find the best tradeoff between bit rate and video fidelity.

Most research efforts have been made to address the challenges of HEVC intra coding merely in algorithm level, while only a few researchers have also attempted to design hardware feasible algorithms. In this dissertation, we investigate to realize a real-time HEVC intra encoder, which is capable of handling video compression of 4K@30fps videos, meanwhile retaining the highest video quality. The entire work involves two aspects described as follows.

From an algorithm point of view, efficient algorithm adaptations need to be proposed to reduce the computational complexity of HEVC intra encoders. The enhanced coding features of HEVC have encouraged a much more complex RDO process, which contains a large number of CU/PU/TU combinations. However, when it comes to hardware implementation, it is cost prohibitive to undergo every possible RDO candidate and CTU partition due to its limited timing, power, and chip area budget [13]. Therefore, the first challenge is to design both efficient and

hardware-friendly video coding algorithms.

From the hardware implementation point of view, an efficient hardware architecture has to be devised to satisfy the throughput requirement of 4K@30fps video compression. In order to accelerate video compression, multiple hardware accelerating techniques have to be exploited, such as pipelining, parallel processing, etc. Except for throughput, other factors also have to be considered in hardware architecture design, like logic consumption, memory bandwidth, clock frequency, etc. For pipeline stages, the throughput of each stage has to be balanced to maximize the usage of computation. To reduce the chip size, logic and memory reuse technique has to be employed. Thus, the second challenge is to design an area, power, and performance optimized hardware architecture of HEVC intra encoder.

1.4 DISSERTATION OUTLINE

In this chapter, we have introduced the background, motivations, and challenges of our research. The rapid growth of video applications, the increasing popularity of HD videos, and the emergence of ultra-high-definition video formats (4K, 8K) have made the demands for superior coding efficiency to H.264 stronger than ever. However, despite the incredible advantages with HEVC standard, its complexity has also increased drastically, making HEVC hardware infeasible without complexity reduction. Most research efforts focused on software optimizations without considering hardware feasibility. To realize HEVC intra encoder in practice, we have to design hardware friendly algorithms to reduce computational complexity as well as an area, power, and performance optimized hardware architecture to sustain highly efficient video compression.

In chapter 2, we will briefly introduce the major functions of video coding and HEVC standard. Then, academic works regarding complexity reduction, rate-distortion optimization,

mode preselection, and CU partitioning will be presented.

In chapter 3, we propose a group-based scheme to parallelize syntax element processing in table-based CABAC rate estimation, which is an essential part of the entire RDO process. The group-based method can significantly improve rate estimation throughput, which is always the bottleneck of RDO. A highly-parallel architecture is sophisticatedly designed to implement the proposed group-based CABAC rate estimator. Experiments show that the proposed rate estimator can provide more accurate rate prediction than existing state-of-the-art approaches, meanwhile satisfying the throughput requirement of rate estimation for 4K video compression @ 30fps.

In chapter 4, we propose efficient algorithm adaptations of HEVC intra encoder and its corresponding hardware architecture design. The proposed algorithm adaptations can drastically reduce the complexity of HEVC intra encoding, meanwhile still retaining excellent video quality. The coding efficiency of proposed algorithms is evaluated and compared with the original HM-15.0 reference software. The hardware architecture is designed to maximize the parallelism of intra prediction with a slight overhead of resource and power consumption. Comparison with state-of-the-art researches is carried out to demonstrate the advantages of our proposals.

In Chapter 5, an enhanced intra encoder is presented based on the performance analysis of the proposed design in chapter 4. The derived luma mode is included in chroma prediction, while it has been excluded in the previous design. Compared with the design in chapter 4, the enhanced intra encoder improves its compression efficiency greatly in worst-case scenarios, making it more reliable. Then, a more balanced prediction block scheduling scheme is proposed to improve the overall throughput of intra encoder. Moreover, in this new design, the operating frequency of rate estimation has been doubled to improve its processing speed in order to

mitigate the throughput decrease caused by derived luma mode inclusion.

In chapter 6, a fully connected layer based neural network (FCLNN) is investigated as an alternative of RDO algorithm to determine intra prediction. Motivated by the superior efficiency of deep learning techniques in potential feature extraction, a two-stage luma mode preselection scheme is designed. The proposed mode preselection scheme aims at reducing the computation complexity of RDO process in HEVC intra encoder.

In chapter 7, the conclusion and contributions of this dissertation are summarized. Future work is also discussed.

CHAPTER 2

HEVC COMPLEXITY ANALYSIS AND LITERATURE REVIEW

The standardization of HEVC aims to provide about 50% more bitrate reduction at the same video quality over the H.264 standard [9]. Developed from H.264, HEVC has made several improvements in various aspects of video coding. For example, HEVC supports a large variety of block sizes in coding unit, prediction unit, and transform unit to improve its coding efficiency in redundancy elimination. Besides, HEVC also introduces new coding tools, such as quad-tree coding structure, residual quad-tree (RQT) transform coding, advanced motion vector prediction (AMVP), refined intra prediction modes, sample adaptive offset, etc. In this dissertation, we only focus on algorithm adaptations and hardware architecture design for intra-frame coding, thus, in this chapter, only intra coding related features will be introduced and analyzed.

Many research efforts have already been made to address the challenges introduced by HEVC standard since its initial release in 2013. A breakdown analysis of rising challenges in HEVC intra-frame coding will be given in section 2.2. In particular, the details of the rate-distortion optimization (RDO) process that contains a series of computation-intensive and time-consuming processes, including prediction, transform, quantization, inverse quantization, inverse transform, rate and distortion estimation, and reconstruction will be presented. State-of-the-art researches conducted to reduce algorithm complexity will be introduced in section 2.3, including prediction mode reduction, CU size fast decision, and simplified rate and distortion models.

2.1 INTRA FRAME CODING IN HEVC

Intra-frame coding is one of the most important parts of video compression. It relies on the detection of spatial redundancy to realize compression. Intra-frame coding can be scheduled independently or used as the starting frame of a GOP (Group of Pictures), which is a pre-defined

structure that contains a series of consecutive frames to be coded as an integral part. Intra-frame demonstrates its advantages over inter frames from several aspects. First, intra frame coding is less computational than inter frame coding, which involves in computation-intensive motion search. Second, intra frame coding of each prediction block relies on reconstructed pixels of its previous coded blocks, which requires on-chip memory to buffer reconstructed pixels. However, compared with inter frame coding that depends on reference frames, intra frame coding requires much less memory resource. Moreover, the hardware implementation cost of intra only encoder is much smaller than encoder with inter frames. The only disadvantage of the intra-only encoder is its lower efficiency in terms of compression ratio and video quality. In video compression, a larger compression ratio indicates a smaller video bit rate.

In HEVC, the coding efficiency of intra encoder has been greatly improved with several major enhancements. For example, large coding, prediction, and transform blocks enable to find more data redundancy during prediction, transformation, and quantization. The increased number of intra prediction modes from 9 to 35 greatly enlarges the searching space for the best prediction pattern, resulting in a more accurate prediction. The quad-tree coding structure allows finding the best CU partitioning of each CTU. In the following sections, we will briefly introduce those coding features.

2.1.1 HEVC Intra Prediction Modes

In intra-frame coding, each prediction block has to first go through intra prediction, which is the first stage in all hybrid block-based video coding. Intra prediction tries to predict the current block with reference pixels that are derived from reconstructed pixels of previously coded blocks. Each prediction block is generated according to the prediction mode, which defines exactly how the pixel values of the prediction block are calculated from reference pixels.

Fig. 2.1 shows two frequently used prediction modes: horizontal mode and vertical mode. Symbol A to symbol I indicate the reference pixels of the current 4×4 PU. In horizontal mode, pixel values of each row are set equal to the reference pixel of the same row, while in vertical mode, pixel values of each column are set equal to the reference pixel of the corresponding column.

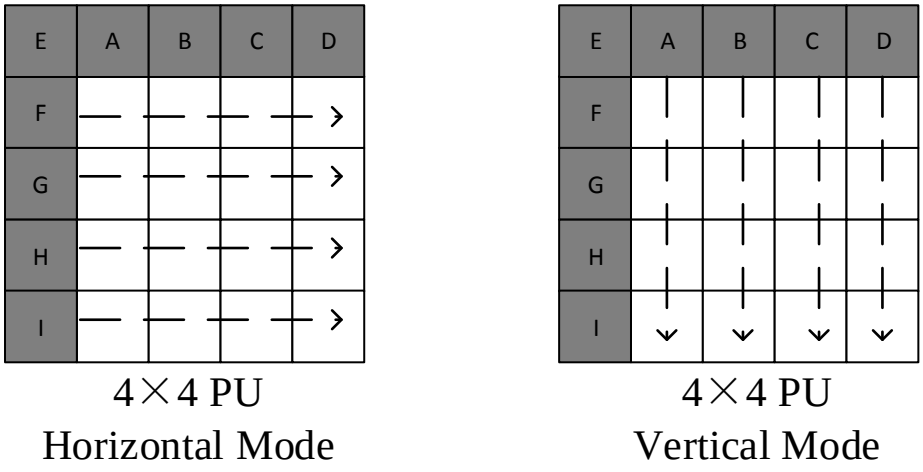


Fig. 2.1. Prediction mode examples: Horizontal Mode and Vertical Mode.

The predicted pixel values of the current block will be further compared with its original pixel values that convey information of interest. A prediction error block, also known as a residual block, is derived by subtracting prediction block from the original block. Therefore, each prediction mode will result in a dedicated residual block, which will be further used for compression instead of original pixels.

The residual block goes through a series of computation processes, including transform, quantization, reconstruction, rate estimation, etc. Rate-Distortion cost is usually adopted as the metric of coding efficiency. Thus, the entire process of intra prediction is trying to find the best prediction mode that brings about the smallest Rate-Distortion cost. With more prediction modes introduced in HEVC, it is more likely to perform better prediction. In HEVC standard, there are a total of 35 prediction modes for luma PUs, including DC, Planar, and 33 directional modes as

shown in Fig. 2.2. Among all 33 directional modes, Horizontal and Vertical modes with index 10 and 26 are most frequently used. For chroma PUs, HEVC introduces one extra mode in addition to four fixed modes (DC, Planar, Horizontal, and Vertical) as in H.264. The extra chroma mode is derived from the corresponding mode of luma PU to enable better prediction in the case of luma and chroma PUs possessing similar texture properties.

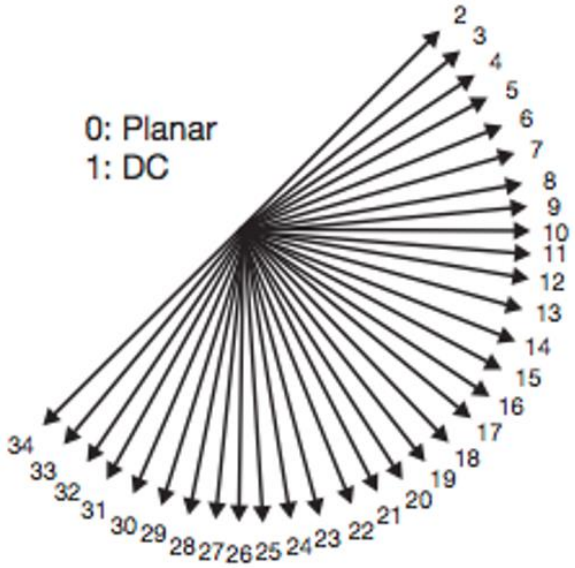


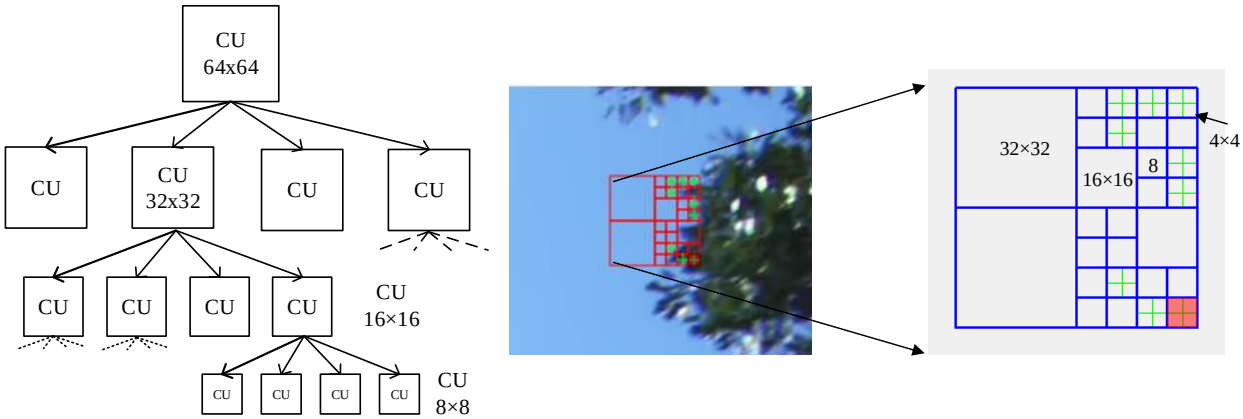
Fig. 2.2. 35 intra prediction modes for Luma PUs in HEVC standard

2.1.2 Coding Block Size and Structure

HEVC introduces the concept of coding tree unit (CTU) to replace macroblock (MB) in H.264 [13]. CTU size can be configured from 16×16 up to 64×64, while MB is fixed to 16×16. Each CTU consists of a certain number of CUs, which is determined by CTU partitioning. The supported CU size in HEVC is from 8×8 up to 64×64. In intra frame coding, a large size CU can be recursively split into four sub-CUs until the smallest allowed CU is reached, which is known as the quad-tree coding structure. Each CU consists of PU and TU. PU conveys prediction related information, including one Luma PB and two Chroma PBs, while TU contains transform

related information, including one Luma TB and two chroma TBs.

Fig. 2.3(a) shows an example of a quad-tree structure of CTU 64×64. With a wide range of block sizes available and quad-tree coding structure, HEVC is able to achieve a higher coding efficiency than H.264 in terms of compression ratio and video quality. Large coding blocks are suitable for smooth texture areas, while small coding blocks show advantages in detail-rich areas as shown in Fig. 2.3(b).



(a) Quad-tree structure (b) CTU partition example

Fig. 2.3. Quad-tree coding structure

Table 2.1 A comparison of major functionalities between HEVC and its predecessor.

Functionality	H.264/AVC	H.265/HEVC
Coding Tree Unit (CTU)	Macroblock (MB) 16×16	Chosen from 16×16, 32×32, and 64×64
Coding Unit (CU) Sizes	Only 16×16	8×8, 16×16, 32×32, and 64×64 Quad-tree structure
Prediction Unit (PU) Sizes	4×4, 8×8, and 16×16	4×4, 8×8, 16×16, 32×32, and 64×64
Prediction Modes	Luma: 9, Chroma: 4	Luma: 35, Chroma: 5
Transform Unit (TU) Sizes	4×4 and 8×8 Only DCT	4×4, 8×8, 16×16, and 32×32 DCT and DST

A comparison of key coding features has been made between HEVC and its predecessor

as shown in Table 2.1. It is obvious that HEVC supports more block sizes of CU, PU, and TU, which ensures a better compression performance.

2.2 RATE-DISTORTION OPTIMIZATION

As we have introduced the major improvements of HEVC over H.264 in the above sections, the process that determines CTU partition and prediction modes is presented in this section. It is also known as rate-distortion optimization (RDO), which consists of a series of computation processes, including prediction, transform, quantization, inverse quantization, inverse transform, reconstruction, rate estimation, and distortion calculation. The goal of RDO is to find the best tradeoff between bitrate and video fidelity. The bitrate is closely related to the compression ratio, while video fidelity reflects the quality loss. To quantify the coding efficiency for comparison purpose, the Lagrangian cost is adopted and calculated by:

$$J = D + \lambda * R \tag{2.1}$$

Where J is the Lagrangian cost, D is distortion, R is bit rate, and λ is the Lagrangian multiplier. Distortion D is computed by the sum of squared errors (SSE) between reconstructed pixels and original pixels. Rate R represents the bit rate of output bitstream. In HEVC reference software, table-based context-adaptive binary arithmetic coding (CABAC) is employed for rate estimation. Since CABAC is a context-based and highly serial process, it is a time-consuming procedure to estimate the bitrate. Lagrangian multiplier λ is an experimentally determined parameter, which can be adaptively changed according to different applications.

Given the Lagrangian cost function (2.1), the RDO process which aims at finding the best PU modes and CTU partition can be performed accordingly. For each luma and chroma PBs, RDO considers all potential prediction modes in a full-search scheme. For example, the RDO process will be executed 35 times to find the best prediction mode of a luma PB with the smallest

Rate-Distortion cost. Moreover, in order to find the best CTU partition, RDO has to be performed for each possible combination of partition. Since HEVC supports a wide range of CU, PU and TU sizes that result in a huge number of CTU partitions, it is infeasible to run RDO for all candidates due to the limited availability of timing and computation in real application scenarios.

2.3 LITERATURE REVIEW

As has been discussed above, HEVC brings about superior compression performance to its predecessor, however, the complex RDO process makes HEVC much more difficult to be commercialized. Therefore, many research efforts have been made recently to address all these challenges in implementing the HEVC standard. Most of the researchers focus on improvement on the software side, while only a few researchers try to build up hardware friendly solutions. There are majorly three aspects in the RDO process, where research efforts are made to reduce HEVC complexities.

The first one is prediction mode reduction, which aims to reduce mode candidates for the RDO process. Authors in [14] proposed the use of Hadamard cost based rough mode decision (RMD) and MPM (most probable mode) modes as RDO candidates. Their experiments are carried out based on HM 1.0 with Intel CPU and achieves 20% and 28% encoding time savings on average in high efficiency and low complexity profiles, respectively. In [15], a progressive rough mode search (pRMS) is proposed. It adopts a 4-step mode searching scheme, which can progressively refine its mode selection from the previous step. As reported in [15], their approach reduces encoding time by 26% on average. Authors in [16] proposed to use an edge-detection based method to establish an edge map, which is further used to select the candidate modes for each prediction block. Additionally, with the edge map, redundant CU sizes can also

be eliminated to reduce the number of partitions. The proposed method achieves encoding time saving by 56.8% at the cost of 2.5% BD-Rate increase.

The second category is the fast CU size decision. As HEVC supports a wide range of CU sizes from 8×8 to 64×64, each pixel will be repeatedly computed to find the best CTU partition. Fast CU size decision aims to remove the redundant CU sizes for the RDO process or select the best CU size without performing the full RDO process. In [17], Leng proposed a combined CU size decision scheme, which contains frame-level CU preselection and CU-level size elimination. In frame-level CU pre-selection, the percentage of adopted CU sizes that have been recorded is being referred to as evidence to skip rarely used CU sizes. In CU size elimination, the neighbor and co-located coding unit information are referred to for removing CU sizes that are unlikely to be chosen. The CU size preselection approach provides 45% encoding time saving as reported in [17]. To reduce the complexity of CU size decision, authors in [18] proposed a fast splitting and pruning method, which contains two steps: early CU splitting decision and early CU pruning decision. For each CU, a Bayes decision rule method is employed for early CU splitting and pruning tests. Statistical parameters used for fast CU partitioning are periodically updated online in order to cope with various statistical characteristics of different video frames. Experimental results show their approach can save 50% encoding time on average with only 0.6% BD-Rate increase. Ma proposed an early CU termination method based on an adaptive energy threshold, which is defined by previous CUs [19]. Along with other complexity reduction algorithms, like mode reduction, the proposed approach saves 30% encoding time.

The third category is the simplified RD model, which is expected to replace the computation-intensive and time-consuming RD model in the RDO stage [20]. For each coding block, distortion calculation involves a series of computational processes from prediction to

reconstruction, while rate estimation relies on a highly dependent CABAC algorithm. To simplify RD cost calculation, the authors in [21] proposed to exploit the transform-domain for RD cost calculation, which means computation processes, like inverse transform, quantization, and rate estimation, can be avoided in RDO process. In their approach, the bit rate is estimated merely based on quantized coefficients along with pre-calculated parameters. The parameters empirically obtained from statistics and curve fitting of previously encoded video frames can be adaptively updated to cope with different video characteristics. And distortion in [21] is measured between transform coefficients and inverse quantized coefficients, without inverse transform and reconstruction. The transform-domain based RD model is reported to save 40% encoding time on average with a negligible performance loss. The authors in [22] proposed a four-pixel-strip based ESAD and quantized coefficients based linear model to perform the RDO process, respectively. The proposed RD model was implemented with TSMC 65nm technology and the achieved area reduction for distortion and rate is 42.2% and 93% respectively. [23] proposed a non-zero coefficient-based method, which estimates rate according to the magnitude. The achieved time-saving is 32% on average with about 8.4% BD-Rate loss.

From the above introduction of state-of-the-art researches, it is noticeable that most of the proposals aimed to reduce the timing cost of software encoders, regardless of the feasibility of their algorithms when it comes to hardware implementations. However, practical video encoders are mostly running on a hardware platform like FPGAs or ASICs, because the hardware is usually a hundred times faster than software solutions. Especially for HEVC standard, due to its intensive computation requirement, a hardware implementation of HEVC encoder is indispensable [24]. Therefore, considering hardware feasibility is with the same importance as complexity reduction in developing efficient HEVC algorithms.

2.4 RESEARCH GOAL

In this dissertation, we focus on the exploration of high-throughput hardware architecture design and efficient algorithm adaptations of HEVC intra encoders.

From an algorithm point of view, we aim to design efficient complexity reduction algorithms, which can relieve the computational burden in the RDO process. Strategies, like mode reduction, CU size early determination, and simple Rate and distortion models, need to be exploited. Meanwhile, the hardware feasibility of proposed algorithms must be taken into account as well as coding efficiency.

From the hardware design perspective, in order to perform real-time video encoding of 4K@30fps, a high-throughput hardware architecture of HEVC intra encoder has to be devised. Also, all proposed algorithm adaptations have to be realized in hardware to maintain consistency between software and hardware. In order to satisfy the throughput requirement, the architecture design should maximize the utilization of parallelism.

CHAPTER 3

PROPOSED TABLE-BASED RATE ESTIMATION

In this chapter, we introduce a highly-parallel hardware architecture design of the table-based CABAC rate model, which aims to improve rate estimation throughput in HEVC intra encoders by employing more parallelism. The proposed rate estimator adopts a table-based context-adaptive binary arithmetic coding (CABAC) bit estimation of HM-15.0 [25]. Modifications have been made to make the CABAC rate model hardware feasible without large compressing efficiency degradation. The algorithm adaptations have been verified in the HM-15.0 reference software. Experiments show a negligible loss on average of 0.005% and 0.0092dB in BD-Rate and BD-PSNR, respectively. To improve rate estimation throughput, a new concept of syntax group has been proposed. All syntax elements that involve in rate estimation have been split into five different groups, where context model dependency does not exist between different groups. Based on syntax groups, a highly-parallel hardware architecture is proposed to parallelize the processing of syntax elements to increase the rate estimation throughput. The proposed hardware architecture was implemented in Verilog and synthesized with FPGA and ASIC. Compared with the state-of-the-art hardware designs for rate estimation in the literature, our design achieves a great improvement in rate accuracy with a slight overhead of chip area and power consumption. To our best knowledge, our approach is the first work to implement the table-based CABAC rate estimator in hardware, achieving high rate estimation accuracy, but maintaining high-throughput by parallelism. The proposed hardware architecture is attractive in time-constrained, high-performance video coding applications.

3.1 INTRODUCTION

As discussed in the previous chapters, HEVC possesses superior compression efficiency

to its predecessor H.264, however, the resultant computational complexity is also tremendous. The main reason is that the enhanced coding features of HEVC lead to an extremely complex RDO process, which aims to make the best trade-off between compression ratio and video quality [26]. The optimization process of RD contains a series of computation tasks such as intra prediction, transform, quantization, inverse quantization, inverse transform, rate estimation, distortion calculation, etc. Even with complexity reduction methods, software implementation of the RDO process is not capable of handling with UHD video compression in real-time scenarios. Therefore, it is definitely a better choice to perform HEVC video encoding on a hardware platform such as FPGA and ASIC. With a good pipelining design, computation-intensive tasks like prediction, transform, quantization, inverse quantization, inverse transform, and distortion estimation can be executed properly with high throughput. However, the CABAC rate estimation is a highly serial process that relies on context modeling and updating. Syntax elements that share context models exist a strong data dependency, which leads to the low throughput of CABAC based rate estimation [27]. Hence, it is of great necessity to develop a high throughput hardware architecture for CABAC based rate estimation.

Lots of research efforts have been made to address those challenges. Most of them have been conducted to develop new rate models to replace the CABAC model. Only a few research works have tried to improve the throughput of the CABAC model, which is one of the purposes of our research. The proposed hardware architecture has the following three aspects of advantages.

First, from an algorithm efficiency perspective, the table-based CABAC rate estimator is a more accurate and reliable method than other alternatives, because it follows the essential computation flow of CABAC entropy encoder that generates the output bitstream. Table-based

CABAC rate estimator and CABAC entropy encoder have the same binarization, context modeling, and context updating. The former that adopts look-up tables to replace binary arithmetic coder (BAC) aims to output the bit number, while the latter performs binary arithmetic coding to generate the bitstream. However, none of the existing designs in the literature implement context-adaptive probability modeling and updating scheme for bitrate estimation. Therefore, the discrepancy between the predicted rate and the real rate is inevitable in those designs. In this dissertation, since we adopt the table-based CABAC rate estimator that matches the context-adaptive probability modeling with the CABAC bitstream encoder, it naturally ensures reliable, consistent, and accurate rate prediction.

Second, from a user convenience point of view, the proposed rate estimator does not rely on preprocessing or offline training which is mandatory in [23, 28 and 29]. For example, the design in [23] requires statistical data collection and curve fitting, which determine the key parameters that are used in the rate prediction. Authors in [29] proposed to utilize a series of probability density functions that define the ratio between bit number and bin number. These functions are established through statistics, which cannot guarantee good accuracy of bit rate estimation for any scenario. Actually, it is more likely that rate estimation accuracy may vary widely depending on video content. In contrast, the proposed CABAC rate estimator fully conforms to the procedure of CABAC entropy coding, so good rate prediction can be ensured and the accuracy may change slightly with video contents.

Third, from a hardware implementation and performance point of view, our proposed hardware architecture is highly efficient and enables high-throughput rate prediction. We propose a three-parallelism rate estimation scheme, which consists of parallel rate estimation of syntax groups of each PB, parallel rate estimation of different prediction modes of each PU, and

parallel rate estimation of different size PUs. First, we propose to divide all syntax elements into five syntax groups. Syntax elements that share context models are assigned to the same group. Thus, context model dependency does not exist between syntax groups, indicating parallel processing of syntax groups is possible. Second, since each PU may have not only one prediction mode in the RDO process even after mode reduction, we propose to instantiate multiple rate estimation instances to maximize rate estimation throughput. Moreover, the proposed hardware architecture also enables parallel rate estimation of different size CUs.

The proposed design is fully conformed to the table-based CABAC rate estimator except ignoring the syntax element "*split_cu_flag*", which results in a negligible coding efficiency degradation. A sophisticated hardware architecture has been designed and realized in Verilog. The hardware design has been evaluated by FPGA and ASIC technologies. The relationship among QP values, PSNR, CU sizes, and the number of required clock cycles for rate estimation have been explored. Compared with existing hardware rate estimators, our design demonstrates a significant advantage in accuracy and reliability, with the overhead of a relatively larger chip area and higher power consumption.

The rest of this chapter is organized as follows. In section 3.2, the RDO process and existing state-of-the-art designs will be briefly reviewed. The details of hardware architecture will be revealed in section 3.3. Experimental results and comparisons are presented in section 3.4. The conclusion is drawn in section 3.5.

3.2 RATE-DISTORTION OPTIMIZATION LITERATURE REVIEW

In major video coding standards, rate-distortion optimization has been widely employed to make the best trade-off between rate and distortion. The superior compression efficiency of HEVC leads to an extremely complex and computation-intensive RDO process. Fig. 3.1 shows

the computation flow of the RDO process, where distortion calculation involves a series of computation processes such as prediction, transform, quantization, inverse quantization, inverse transform, and reconstruction. On the other hand, rate estimation is performed on quantized coefficients through the CABAC algorithm, where syntax elements are processed serially due to context model dependency. The resultant low-throughput rate estimation further impedes the overall processing speed of RDO. As been introduced previously, the Lagrangian cost function J_{RD} is adopted to assess the coding efficiency of each candidate.

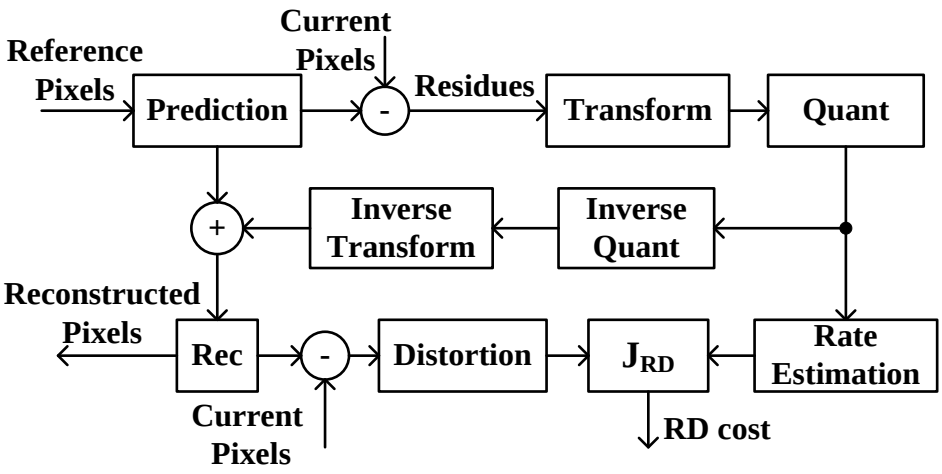


Fig. 3.1. Computation flow of RDO process

3.2.1 Existing Hardware Architecture for Rate Estimation

Directly implementing CABAC entropy encoder for rate estimation is inefficient due to the inherent dependency during context modeling and binary arithmetic coding. Thus, a table-based CABAC rate estimation scheme is proposed to evaluate the rate without binary arithmetic coding to reduce dependency. However, context modeling between syntax elements is still highly dependent on the table-based CABAC rate estimator.

So far, very few hardware solutions of rate estimation have been presented in literature and their computational diagrams are illustrated in Fig. 3.2(a)-(c). In Fig. 3.2(a), non-zero

quantized coefficients are directly employed to develop new rate models to replace CABAC entropy encoder. In Fig. 3.2(b), bin number is adopted as the rate number. In Fig. 3.2(c), a table-based context-fixed binary arithmetic coding (CFBAC) scheme is proposed. Fig. 3.2(d) shows the table-based CABAC bit rate counting algorithm, which is adopted in our dissertation.

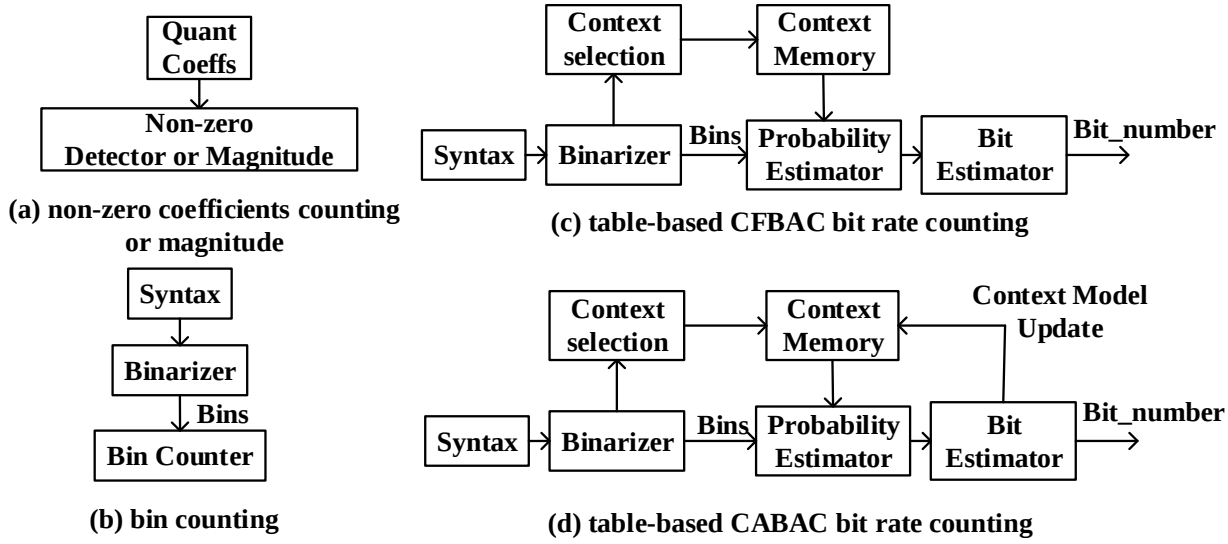


Fig. 3.2. Computational diagrams of existing rate estimation algorithms

Table 3.1 shows a comparison of existing rate estimation algorithms that have been implemented in hardware. Non-zero coefficient counting is proposed based on the assumption that transformed residuals follow a Laplacian probability distribution, the bit rate is approximated by counting non-zero coefficients after quantization. Based on the magnitude of non-zero coefficients, new rate models have been developed for low complexity. These models are based on statistical analysis. Therefore, a preprocessing stage or offline training is required to perform data collection and curve fitting. However, statistical methods cannot guarantee a good generalization of the derived rate model due to the unpredictable variety of video contents [30]. Bin counting method uses bin number directly as bit rate number without considering the influence of entropy coding. In CFBAC, context models are fixed, thus syntax elements can be

processed in parallel without the conflict risk in context modeling. Among all the above rate estimation algorithms, CFBAC achieves the best compression performance when compared with the CABAC algorithm. As shown in Table 3.1, CFBAC still has a 1.14% BD-Rate increase, because the redundancy between syntax elements with the same context model has been ignored.

Table 3.1 Comparison of Rate Estimation Algorithm in HEVC.

Rate Estimation Algorithm	**ΔBD-Rate [%] (vs. CABAC)**
Non-zero Quantized Coefficients Counting [21]	N/A
Magnitude of Non-zero Quantized Coefficients [22, 23, 28, 31]	6.27 [22] 4.53 [23]
Bin Counting [29]	2.65 [24]
Table-based CFBAC Bit Counting [32]	1.14 [24]
Table-based CABAC Bit Counting [25]	-0.13 [24]

In this dissertation, we adopt table-based CABAC bit rate counting, which shows superior coding performance than existing algorithms. Compared with CABAC entropy encoder, the table-based CABAC rate estimator achieves a slight decrease of 0.13% in BD-Rate as shown in Table 3.1. Due to the negligible difference, the table-based CABAC rate estimator is the best alternative of CABAC entropy encoder for rate estimation. The advantage of table-based CABAC rate estimator is using look-up tables to replace binary arithmetic encoder to reduce the inherent dependency in bitstream generation, which is not unnecessary in rate estimation. It has also been adopted in HEVC reference software. However, the throughput of the table-based CABAC algorithm is still limited by existing dependency during context modeling and updating. Thus, it is still a challenge to develop a high throughput hardware architecture of table-based CABAC rate estimator, while retaining its superior compression performance.

The authors in [33] proposed to process 2 syntax elements (i.e., coeff_abs_level_greater1 _flag and sig_coeff_flag) in parallel, while the rest of 14 syntax elements are still processed in a serial manner. With a 0.5% of bit rate overhead, CABAC rate estimation is accelerated. Yet, the potential of parallelism in CABAC rate estimation is not fully investigated.

From the above discussion, it is apparent that these existing hardware architectures in literature have not fully explored the potential parallelism of the table-based CABAC rate estimation. Therefore, it is necessary to develop an efficient hardware architecture for table-based CABAC bit estimation algorithm. This new architecture may exhibit good rate estimation accuracy, meanwhile providing high throughput processing.

3.3 PROPOSED HIGHLY-PARALLEL HARDWARE ARCHITECTURE

3.3.1 Design Methodology

Fig. 3.3 shows the computational diagram of CABAC entropy encoder. The only difference between table-based CABAC rate estimator in Fig. 3.2.1(d) and CABAC entropy encoder is that binary arithmetic encoder is replaced by look-up tables in rate estimator.

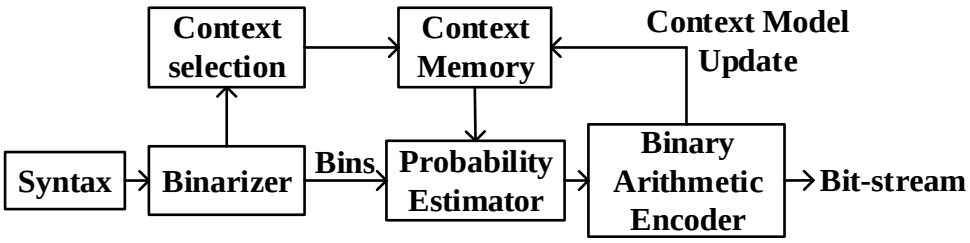


Fig. 3.3. Computational diagram of CABAC entropy encoder.

Since output bitstream is not required in rate estimation, the table-based CABAC algorithm employs three look-up tables to perform context updating and rate prediction. The fractional bit number of each bin is accumulated to provide the total number of bits in rate estimation. Since the binary arithmetic encoder is no longer used, it is possible to process syntax

elements in a more flexible order in rate estimation. Moreover, syntax elements that do not share context models can be processed independently and simultaneously. Inspired by this observation, we propose to divide all related syntax elements into 5 syntax groups, each of which can be processed independently. Syntax elements that require the same context models are grouped together. Therefore, 5 syntax groups can be processed in parallel without context model conflictions and the throughput of rate estimation is improved drastically. A group-based syntax processing hardware architecture of rate estimation is proposed. The proposed design is expected to achieve substantial improvement in rate estimation accuracy and throughput with the overhead of a relatively larger chip area and higher power consumption.

Table 3.2 Group Based Syntax Elements Division.

Syntax Group	Syntax Elements for Luma Rate Estimation	Number of Context Models
A	prev_intra_luma_pred _flag	1
	mpm_idx	0
	rem_intra_luma_pred_mode	0
	part_mode	1
	cbf_luma	2
	transform_skip_flag	1
B	last_sig_coeff_x_prefix	15
	last_sig_coeff_y_prefix	15
	last_sig_coeff_x_suffix	0
	last_sig_coeff_y_suffix	0
C	coeff_abs_level_greater1_flag	16
	coeff_abs_level_greater2_flag	4
	coded_sub_block_flag	2
D	sig_coeff_flag	27
E	coeff_sign_flag	0
	coeff_abs_level_remaining	0
Total	16 syntax elements	84

Table 3.2 shows the results of syntax grouping of 16 syntax elements and the total number of context models involved in rate estimation of intra luma blocks is 84. Group A contains prediction information related syntax elements. Group B includes the last significant

coefficient position related syntax elements. "*coeff_abs_level_greater1_flag*", "*coeff_abs_level_greater2_flag*" and "*coded_sub_block_flag*" are contained in Group C. Group D includes "*sig_coeff_flag*". Group E only contains bypass coded related syntax elements. Each kind of syntax element in Table 3.2 may contain multiple syntax elements. For example, for the only syntax element "*sig_coeff_flag*" in group D, it may contain up to 16 flags according to different 4×4 sub blocks of the quantized coefficient blocks.

In our work, we propose to parallelize the processing of 5 syntax groups to improve the rate estimation throughput of each prediction block. Moreover, we also propose to implement multiple rate estimator instances for more parallelism. Each instance is assigned to estimate the rate of a candidate prediction block of a PU, which contains multiple mode candidates for RDO. All candidate prediction blocks of the same PU share the same initial context models that have been stored in a global context model buffer, which ensures that each prediction block has the correct initial context model values.

Multiple rate instances improve the overall throughput of rate estimation in the RDO process, however, the risk of context model update conflict should be avoided. The reason for potential conflict is because the global context models are shared among multiple rate instances. An updating of a global context model of one rate instance may lead to incorrect context modeling of another rate instance. To address this challenge, we propose to localize context models as shown in Fig. 3.4. The mechanism of our approach is described as follows.

Before starting rate estimation, all rate estimation instances of the same PU need to load context models from the global buffer into their internal register arrays. This operation is called the context model localization. Then, during rate estimation, local context models inside each instance will be updated accordingly, while the global context models remain unchanged. Since

each rate estimation instance has its local context model buffer, the conflicts of context modeling between different instances can be avoided. Once the rate estimation is complete in all instances, PU mode decision is made. Based on the decision result, the derived local context models that correspond to the best PU mode in a rate estimation instance are selected to update the global context models. In this way, our proposed procedures of context model loading and updating prevent potential context model conflict among multiple rate estimator instances of the same PU. For multiple rate estimation instances of different size CUs, we propose a similar procedure, where the 4-level global context model buffer is exploited. Each level of context models corresponds to a dedicated CU size from 64×64 to 8×8.

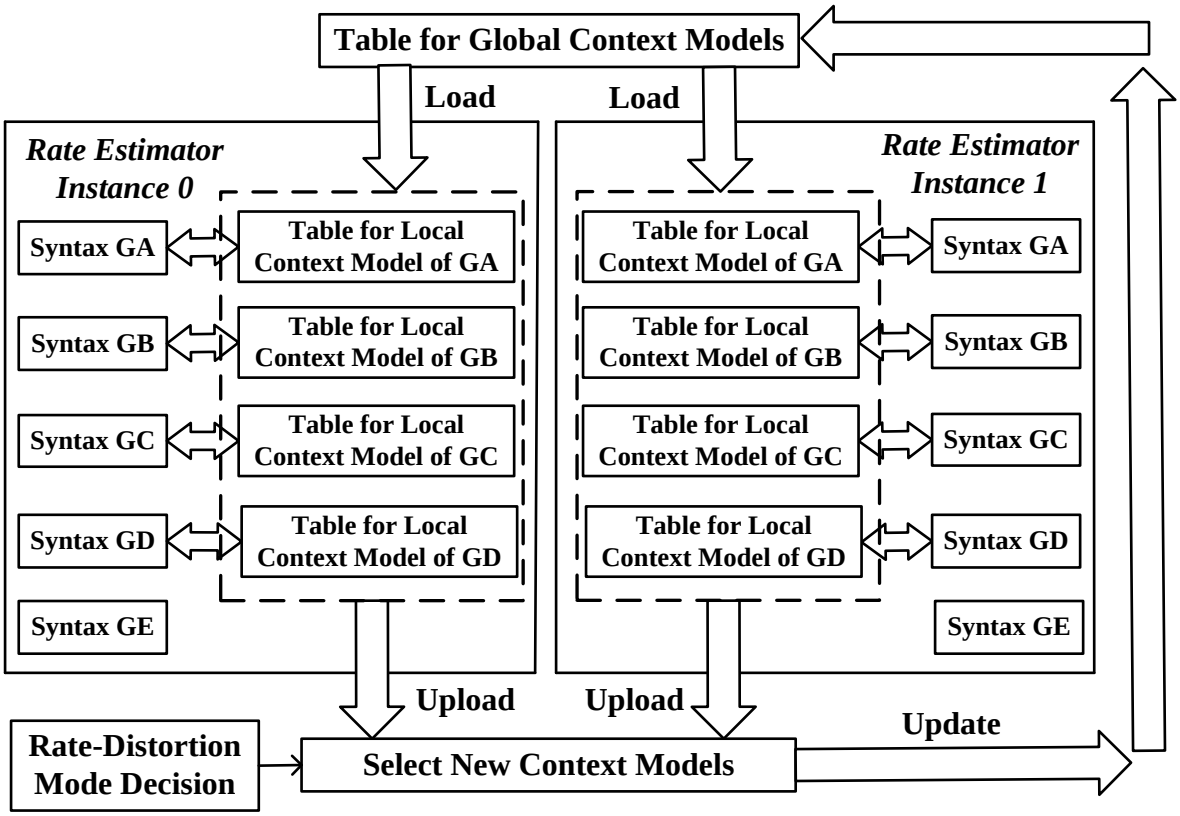


Fig. 3.4. Proposed scheme of context model loading and updating.

3.3.2 Overview of Proposed Hardware Architecture

Fig. 3.5 demonstrates the proposed hardware architecture of the table-based CABAC rate

estimator. The proposed design enables parallel processing of 5 syntax groups independently as shown in the figure. In total, it contains 9 major modules, including Main Controller, Coefficients Loading Controller, Coefficients Preprocessor, Fractional Accumulator, and five syntax group (GA-GE) processors. The operating mechanism of the proposed rate estimator will be described in this section.

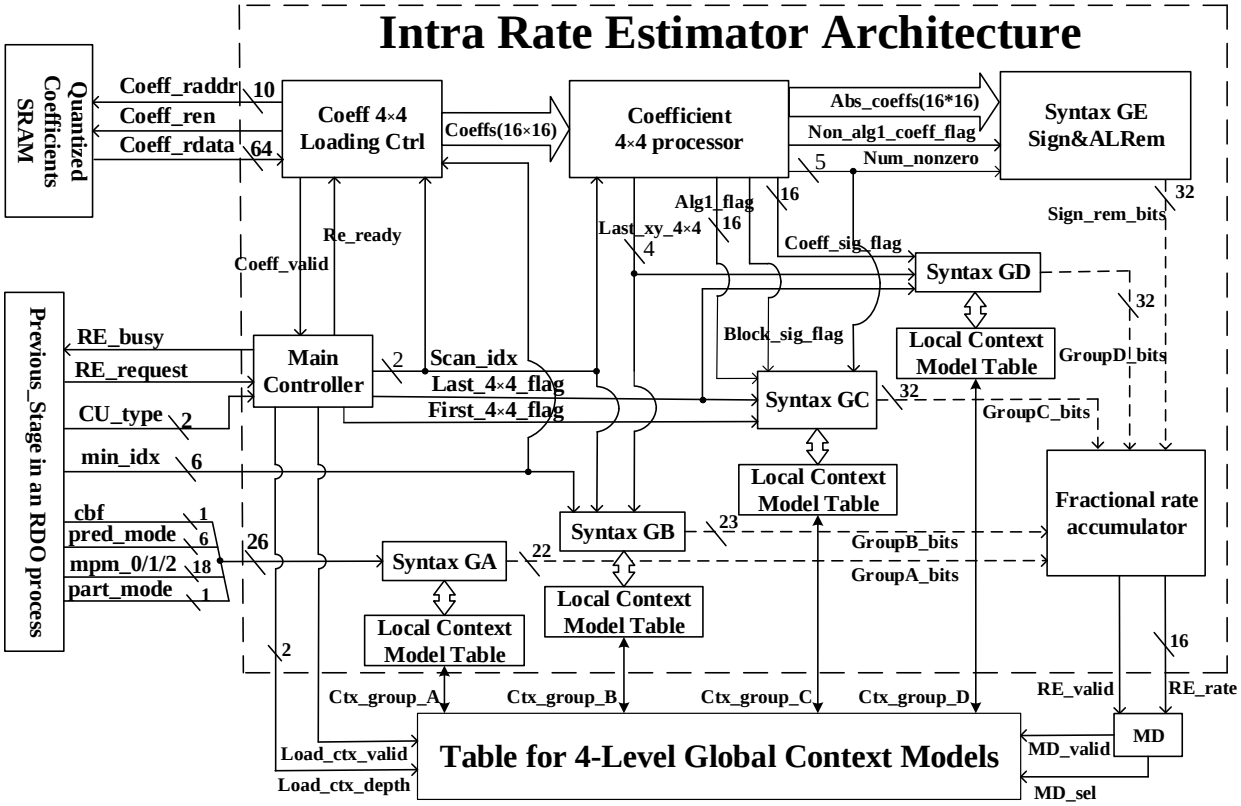


Fig. 3.5. Proposed highly-parallel hardware architecture of a table-based CABAC rate estimator.

Before the beginning of rate estimation of a certain PB, Main Controller (MC) monitors the status of the RDO process. When the quantization process of the PB is finished, a request of rate estimation will be sent to the Main Controller to start rate estimation. Prediction related signals of this PB used in rate estimation will be read as input signals from previous RDO processes. Since the proposed rate estimation algorithm is performed on 4×4 sub blocks, quantized coefficients will be split into sub blocks and read by Coefficients Loading Controller

from on-chip 64-bit-width SRAM. Each sub block contains 16 quantized coefficients, each of which is 16-bit width. The loading of a sub block takes 4 clock cycles with 4 coefficients read per clock cycle. FIFO (first-in-first-out) is adopted to hold the loaded coefficients. Once the sub block loading is completed, all 16 coefficients stored in FIFO will be sent to Coefficient Processor to generate syntax elements. Then, syntax elements will be split into 5 groups and each group will be processed in the dedicated syntax processor as shown in Fig. 3.5. Each syntax processor will perform rate estimation independently and five fractional bit numbers be will accumulated to form the total bit number. When the total bit rate is derived, the control signals "*RE_valid*" and "*RE_rate*" will notify the mode decision block. Based on the "*MD_sel*" signal, the global context models will be updated.

The proposed hardware architecture maximizes the irrelevance between syntax groups and parallelizes the processing of syntax elements. It supports rate estimation of all CU sizes from 64×64 to 8×8 and all TU sizes from 32×32 to 4×4. The PU size is set equal to CU size, except for 8×8 CUs that are associated with either 8×8 PUs or four 4×4 PUs. The TU size is set equal to PU size, except for 64×64 PUs that utilize four 32×32 TUs. The rate estimation of a CU is estimated according to its TUs. If a CU contains multiple TUs, such as an 8×8 CU with a partition mode *N*×*N*, the results of four sub-TUs are added as the total rate of this CU. Rate estimation of a TU is performed on the basis of 4×4 sub-blocks according to its scan direction.

In this design, the pipeline technique is not widely utilized to optimize hardware throughput due to two reasons. First, unlike these coefficients-based or bin-counting-based rate estimators in [12-17] that purely rely on combinational logic gates to estimate rate, the proposed table-based CABAC rate estimator calculates rate values mainly depending on look-up tables (LUTs) of syntax contexts. In order to maximize the throughput, in the proposed rate estimator,

syntax elements are classified into 5 independent groups for parallel processing. In our design, the critical path involves three steps (*i.e.*, context modeling, LUT access, and context model updating). According to the algorithm, adjacent bins often rely on the same context model for rate estimation. Thus, in order to process one bin per cycle and to avoid context conflicts, context models should be used and updated within one clock cycle. Even though a pipelined design helps to shorten the critical path, it also leads to potential conflicts of context models between adjacent bins and causes inaccuracy in rate estimation. Second, synthesis results with TSMC 90nm technology show our proposed non-pipelined hardware implementation is capable of running at a clock frequency up to 640 MHz, which is enough to process a video format of 3840×2160 @ 30fps. Since the non-pipelined design meets the system requirement, there is no need to apply the pipeline technique to further improve the throughput.

Note the syntax element *"split_cu_flag"* is defined in the CABAC algorithm. The bit rate of this syntax element is required when four sub-CUs are compared with a larger CU in the same region. If four sub-CUs are chosen, this syntax element is 1. Otherwise, it is 0. This syntax element is not involved in rate-distortion optimization among various prediction modes of a given CU size. In this proposed architecture, this syntax element is intentionally omitted for reduced design complexity and hardware cost. This choice will be validated by experimental results in section 3.4.

3.3.3 Input Signals and Interface

Input signals (*"CU_size"*, *"min_idx"*, *"cbf"*, *"pred_mode"*, *"mpm_0/1/2"* and *"part_mode"*) feeding rate estimator are obtained from the previous stage in an RDO process. *"CU_size"* indicates current CU size. *"min_idx"* and *"cbf"* are generated after quantization step. *"min_idx"* is the index of last 4×4 block, which contains non-zero coefficients according to its

sub block scanning method. *"cbf"* (coded block flag) indicates if the current TU contains non-zero coefficients or not. *"pred_mode"* indicates the prediction mode of current PU. *"mpm_0/1/2"* indicates the most probable modes (MPMs) derived from neighboring PUs. When a TU is larger than 8×8, only diagonal scan is used. For 8×8 or 4×4 TUs, intra prediction mode determines the scan direction. Specifically, the vertical scan is assigned to the prediction modes from 6 to 14. The horizontal scan is assigned to the prediction modes from 22 to 30. And diagonal scan is applied to the other prediction modes.

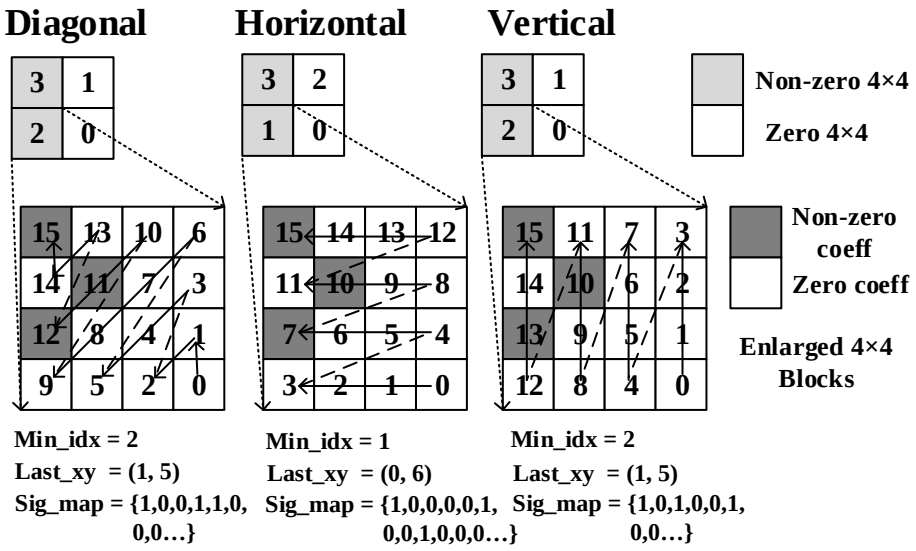


Fig. 3.6. Syntax elements derivation with different scan methods for an 8×8 TU.

Fig. 3.6 illustrates an example of 8×8 TU, where three scan methods lead to different *"min_idx"* values. For an 8×8 quantized coefficient matrix, horizontal scan makes *"min_idx"* to 1, while *"min_idx"* is 2 for diagonal and vertical scan methods. Since the scanning method causes the coefficients in the last sub block to be scanned in different orders, the position of the last significant non-zero coefficient is not identical as depicted in Fig. 3.6. For example, the last coefficient corresponds to index 11 in a diagonal scanning, while it corresponds to index 7 in a horizontal scanning. As a result, significance maps that contain these non-zero coefficients along

the scanning path are different. Since this 8×8 TU in Fig. 3.6 contains non-zero coefficients, in this example, the value of *"cbf"* is 1. Otherwise, the zero value of *"cbf"* indicates only syntax group *A* is required for rate estimation.

3.3.4 Coefficient Loading and Processing

The performance of our proposed rate estimator is slightly affected by the latency of sub block loading. Thus, we propose to use a FIFO to preload several sub blocks to minimize the latency between sub blocks. Fig. 3.7 shows the block diagram of the proposed coefficients 4×4 loading controller, which is the interface with the external quantized coefficients SRAM. In Fig. 3.7, a FIFO sub block buffer (size 4×256) is implemented to store four sub blocks to eliminate the latency caused by coefficient loading between the processing of two adjacent 4×4 sub blocks. Due to the use of FIFO, regardless of the status of rate estimator, loading of 4×4 sub blocks from quantized coefficients SRAM will not stop until the FIFO is full. In addition, as long as the FIFO is not empty, the coefficient 4×4 processor can fetch and process 16 coefficients of one 4×4 sub block from this FIFO. As a result, the initial latency of coefficient process for the first 4×4 sub-block is 6 clock cycles (*i.e.*, 4 for SRAM reading access, 1 for FIFO writing access, 1 for FIFO reading access). Yet, this latency is only 1 clock cycle (i.e., FIFO reading access) for subsequent sub blocks. Compared with the design in [29] which uses 1K byte SRAM for coefficients loading, this proposed architecture adopts about 0.2K byte registers for each rate estimation instance. In fact, loading quantized coefficients to the coefficient loading controller is independent of passing them to the coefficients processor. When the coefficient 4×4 processor sends a "Re_ready" signal to request reading the FIFO, if the FIFO is not empty, one 1×256 data will be loaded into the coefficient 4×4 processor during the next clock cycle. Then, syntax elements and related variables start to generate. For example, syntax element

"coded_sub_block_flag" is derived from checking *"sig_coeff_flag"* of these 16 coefficients. *"last_x_4×4"* and *"last_y_4×4"* indicate the scan position of first non-zero coefficient in this 4×4 block. *"last_x_4×4"*, *"last_y_4×4"* and *"min_idx"* generate *"last_x"* and *"last_y"*, which refer to the scan position of first non-zero coefficient in the current TB. Syntax elements in group *B* are determined by a binarization scheme of *"last_x"* and *"last_y"*.

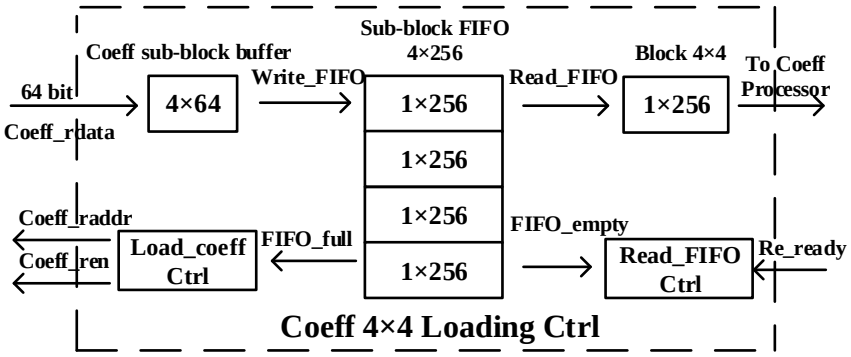


Fig. 3.7. Block diagram of proposed coefficients 4×4 loading controller.

3.3.5 Syntax Group Based Rate Estimation

This section describes how to estimate the fractional rate of each syntax group. As all related syntax elements are assigned to five independent groups, the total bit rate is the sum of five fractional rates (*i.e.*, *GroupA_bits*, *GroupB_bits*, *GroupC_bits*, *GroupD_bits* and *Sign_rem_bits* in Fig. 3.5). The procedure of fractional rate estimation is the same as the table-based CABAC bit rate estimation algorithm in HEVC reference software, including binarization, context modeling, rate estimation, and context updating. As been mentioned earlier, five syntax element groups are processed in parallel without following a defined syntax processing order.

Fig. 3.8 plots the block diagram of proposed bit rate estimator for syntax groups from *A* to *D*. There are three look-up tables (*i.e.*, *MPS_LUT*, *LPS_LUT*, *Entropy_Bits_LUT*), and each of them consists of 128 storage elements. The predicted most probable symbol (*MPS*) is stored as the lowest bit of the current context model, which is a 7-bit binary data. The value of MPS is

compared with the current bin to be coded, and generates a control signal *"Is_MPS"*. Based on *"Is_MPS"* and the current context model, new context model from either *MPS_LUT* or *LPS_LUT* is selected and then updated to the Local Context Models. According to the current bin value and its corresponding context model, the fractional rate of this bin is determined through the look-up table *Entropy_Bits_LUT*. Each fractional rate value is an 18-bit fixed-point number, the highest 3 bits of which represent integer numbers. The number of bypass coded bins exactly represents the rate, therefore, a bypass bin counter is used to calculate bin number. The total rate of a syntax group is accumulated by a *Bits_ACC* block, which adds up all fractional rates of regular bins and bypass bins.

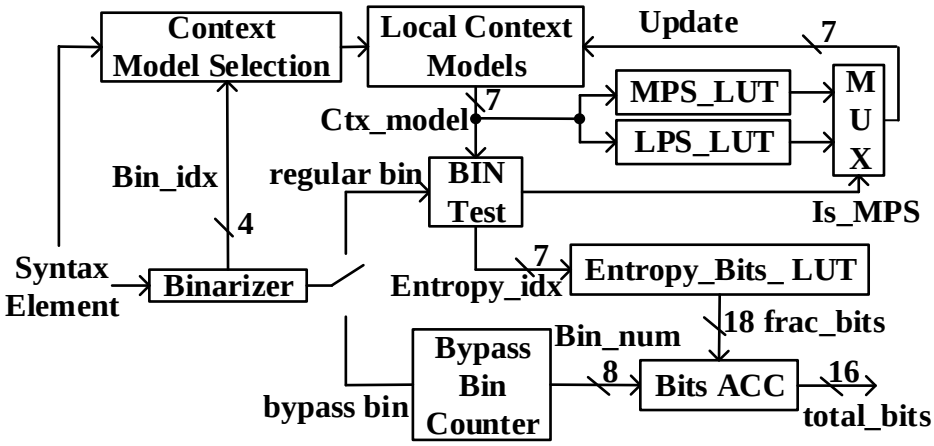


Fig. 3.8. Block diagram of proposed rate estimator for syntax group A-D.

Syntax group *E* consists of two syntax elements *"coeff_sign_flag"* and *"coeff_abs_level_remaining"*. Both of them contain bypass only bins. Therefore, local context models are not required for syntax group *E*. As bit rate of syntax group *E* is equal to the number of bins generated by its syntax elements, a bin counter is implemented to calculate the rate of syntax group *E*. The number of bins for the syntax element "*coeff_sign_flag*" is calculated by coefficient processor. Bin counting of the syntax element "*coeff_abs_level_remaining*" is more critical due to its complex binarization procedure.

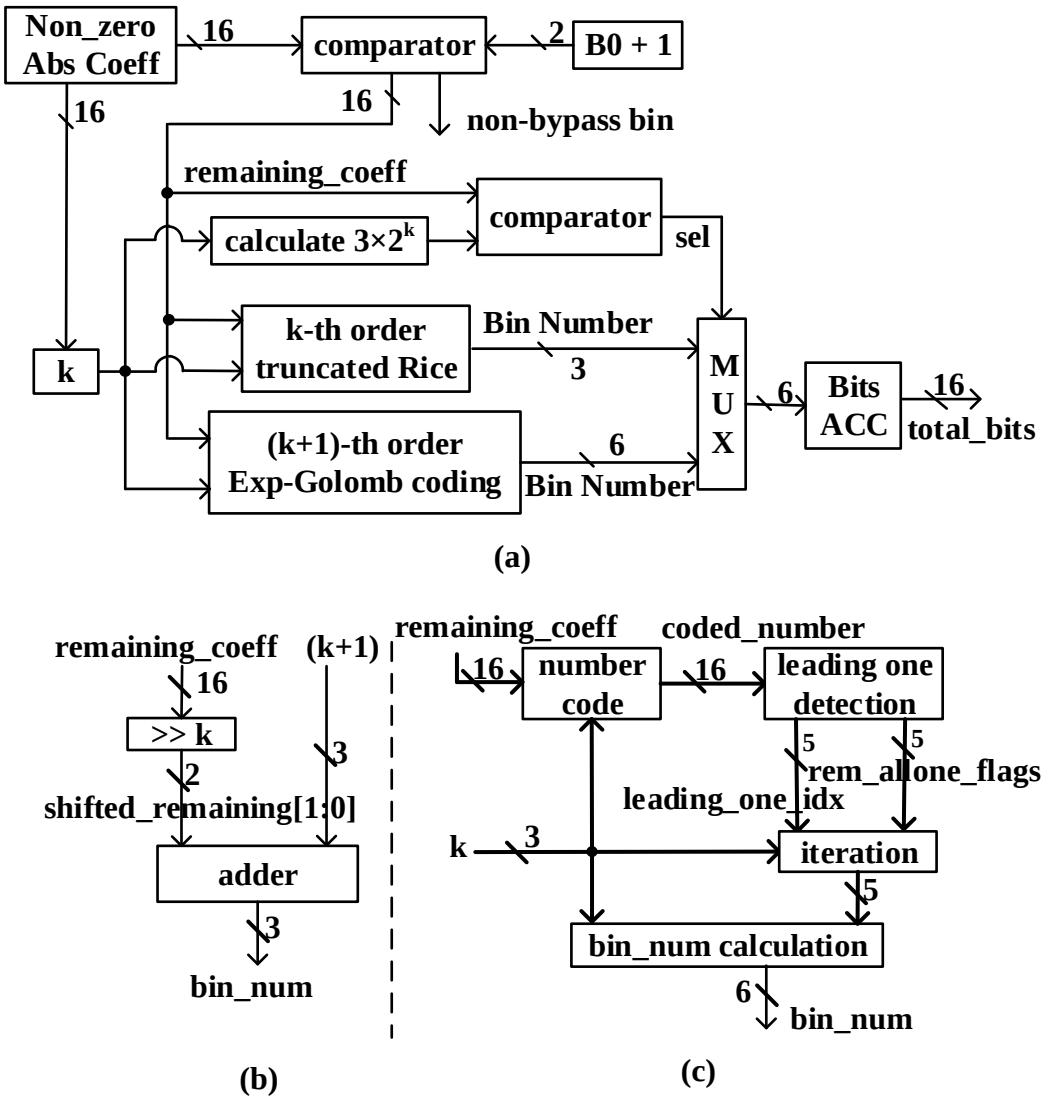


Fig. 3.9. Block diagrams of (a) rate estimator of "*coeff_abs_level_remaining*", (b) k-th order truncated Rice coding, (c) (k+1)-th oder Exp-Golomb coding.

Fig. 3.9(a) illustrates the rate estimator diagram for *"coeff_abs_level_remaining"*, which involves binarization processes of *k-th* order truncated Rice coding in Fig. 3.9(b) and *(k+1)-th* order Exp_Golomb coding in Fig. 3.9(c). Bit rate estimation of *"coeff_abs_level_remaining"* starts with the generation of 16-bit signal *"remaining_coeff"*, which is obtained by comparing non-zero absolute coefficients with *B0+1*. The initial value of parameter *B0* is 2, and becomes to 1 at the first time when the absolute coefficient is larger than 1. Then, *B0* turns to 0 after

processing non-zero coefficients by eight times. If the absolute coefficient is smaller than *B0+1*, no bypass bin is generated. Otherwise, bypass bin number of current syntax "*coeff_abs_level_remaining*" is evaluated based on two arithmetic coding algorithms: *k-th* order truncated Rice and *(k+1)-th* order Exp-Golomb. The value of *"remaining_coeff"* is compared with a constant value 3×2^k, then a resultant selection signal *"sel"* is generated for the multiplexer. Binarization result either from *k-th* order truncated Rice coding or *(k+1)-th* order Exp-Golomb coding is selected and passed to the rate accumulator. The parameter *k* is initially set to 0, and later may be updated to the minimum value of *k+1* and 4. In Fig. 3.9(a), the bin number estimation takes two clock cycles. *"remaining_coeff"* and bin number are calculated in the first and second clock cycles, respectively. So this two-step process is pipelined to achieve a throughput of one coefficient per clock cycle.

Table 3.3 Iteration number calculation table

"leading_one_idx"	Iteration number
<k+1	0
=k+1	1
>k+1	Depend on "rem_allone_flags"

Fig. 3.9(b) illustrates the proposed block diagram of *k-th* order truncated Rice coding. *"remaining_coeff"* is shifted right by *k* bits, then the lowest two bits are sent to a 3-bit adder. Bin number is the output of this adder. Fig. 3.9(c) depicts the block diagram of *(k+1)-th* order Exp-Golomb coding. A 16-bit signal "*coded_number"* is generated and sent to detect leading one. The resultant outputs are two 5-bit signals *"leading_one_idx"* and *"rem_allone_flags"*. Each bit of *"rem_allone_flags"* is obtained by checking every bit of *"coded_number"* from the index (*"leading_one_idx"–1*) to index *k*. As shown in Table 3.3, *"iteration_number"* and *"rem_allone_flags"* depend on *k*. The total number of bins is derived by the equation *"iteration_num"*×*2+4+k*.

3.3.6 Syntax Processing Order and Timing Diagram

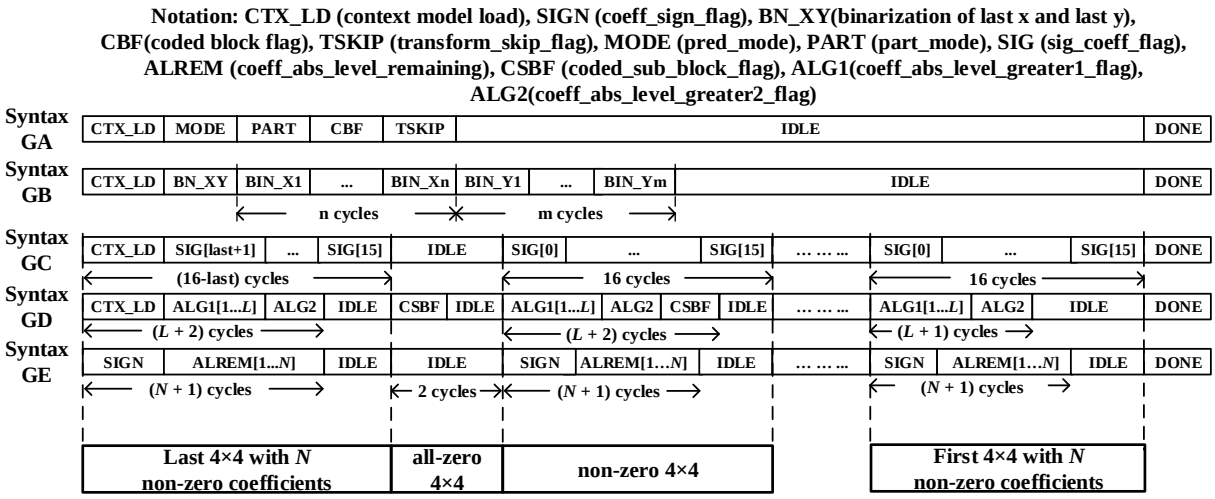


Fig. 3.10. Syntax processing order and timing diagram of the proposed syntax processors.

Fig. 3.10 illustrates the syntax processing order and the timing diagram of five syntax element groups in the proposed rate estimator. Processing of syntax group A and B is performed once for each TB, while processing of syntax group C to E is triggered according to 4×4 sub blocks as shown in Fig. 3.10. Context model loading (CTX_LD) is performed at the beginning of CU rate estimation. Due to a large number of context models and limited SRAM access bandwidth, it takes a lot of clock cycles to load context models from SRAMs and update context models back to SRAMs. Therefore, instead of SRAMs, register arrays are selected as context model buffers, because of its short access time and high-bandwidth for enabling parallel data access from multiple rate estimator instances. In our design, 1 clock cycle is enough to load or update these global context models from or to local register arrays. As syntax group E does not rely on context models, its rate estimation starts with the syntax "*coeff_sign_flag*". The details of the syntax processing order and timing diagram are described below.

In group A, three syntax elements "*part_mode*", "*transform_skip_flag*" and "*cbf_luma*" are regularly coded and each contains one bin. "*transform_skip_flag*" is only for 4×4 TUs in the

TSKIP process. If "*pred_mode*" matches one of three MPM modes, "*prev_intra_luma_pred_flag*" is 1, and "*mpm_idx*" is equal to the index of matched MPM mode. "*prev_intra_luma_pred_flag*" is regularly coded in the MODE process. "*mpm_idx*" and "*rem_intra_luma_pred_mode*" are bypass coded. Binarization of "*mpm_idx*" is performed by the truncated unary. The fixed-length binarization is applied to "*rem_intra_luma_pred_mode*". Bin number of "*mpm_idx*" or "*rem_intra_luma_pred_mode*" is calculated after binarization. The entire processing time for group A is less than 6 clock cycles.

In syntax group *B*, binarization process BN_XY of (*last_x*, *last_y*) is performed after CTX_LD. The signals *"last_x"* and *"last_y",* determined by *"min_idx"* and *"Last_xy_4×4"*, indicate the position of last non-zero coefficient in a TU. Four syntax elements are derived from *"last_x"* and *"last_y"*. *"last_sig_coeff_x_prefix"* and *"last_sig_coeff_y_prefix"* that specify the prefixes of the column and row positions of the last non-zero coefficient contain regular coded bins, while *"last_sig_coeff_x_suffix"* and *"last_sig_coeff_y_suffix"* that specify the suffixes of the column and row positions contain bypass coded bins. Truncated unary binarization is applied to prefix syntax elements, while fixed-length binarization is used for suffix syntax elements. Assume there are *n* regular bins of syntax *"last_sig_coeff_x_prefix"* and *m* regular bins of syntax *"last_sig_coeff_y_prefix"* in Fig. 3.10. Then, according to the bin processing order, the total number of required clock cycles is *(m+n+2)*.

Syntax group *C* contains the regular coded syntax element *"sig_coeff_flag"*. Syntax group *D* contains regular coded syntax elements *"coeff_abs_level_greater1_flag" (ALG1)*, *"coeff_abs_level_greater2_flag" (ALG2)*, and *"coded_sub_block_flag" (CSBF)*. Syntax group *E* only contains bypass coded syntax *"coeff_sign_flag"* and *"coeff_abs_level_remaining" (ALRem)*. Fig. 3.10 shows four different sub block cases of rate estimation for syntax groups *C*-

E. In each sub block, *N* is defined as the number of non-zero coefficients of current 4×4 block and *L* is the maximum number of *"ALG1"* syntax elements to process. *L* is set equal to the smaller value of *N* and 8.

1) Last 4×4 sub block: It is the first sub block to process in rate estimation. In syntax group C, "*sig_coeff_flag*" of the first non-zero coefficient along the scanning path is ignored. Processing of "*sig_coeff_flag*" starts from the next scan position. In syntax group D, processing of "*ALG2*" is performed when there exists an absolute coefficient larger than 1 in this sub block. If this coefficient is also larger than 2, the bin value of "*ALG2*" is 1. The processing of the syntax element "*coded_sub_block_flag*" is ignored for this sub block according to the CABAC rate estimation algorithm. In syntax group E, the processing of "*coeff_sign_flag*" starts immediately. As "*coeff_sign_flag*" is bypass coded, the resulting rate is equal to the number of non-zero coefficients. Each non-zero coefficient is examined by the ALREM process, which performs binarization of the syntax element "*ALRem*". The total number of clock cycles required in Group E is determined by the number of non-zero coefficients. N is the required number of bits to compress "*coeff_sign_flag*". Rate estimation of "*ALRem*" is carried out by the proposed scheme in Fig. 3.10. The required total number of clock cycles is determined by the scan position of the first non-zero coefficient in this block.

2) All-zero 4×4 sub block: All coefficients in this block are zeros. In this case, the only syntax "*coded_sub_block_flag*" with value 0 is processed within 2 clock cycles for all groups C-E. It takes 2 clock cycles to process each all-zero sub block.

3) Non-zero 4×4 sub block: This block contains at least one non-zero coefficient. According to the scan order, it is neither the last 4×4 block nor the first 4×4 block. There are 16 regular coded bins of syntax "*sig_coeff_flag*" to process, so it takes 16 clock cycles to process in

group C. In group D, the process is almost the same as that in the last 4×4 sub block, except a value 1 for "*coded_sub_block_flag*". The number of required clock cycles in group D is determined by L and "*ALG2*". In group E, the required number of clock cycles is determined by (k+1).

4) First 4×4 sub block: It is the final sub block to process. Processing this block is similar to that with the non-zero 4×4 sub block, except the absence of "*coded_sub_block_flag*" in group D. The required numbers of clock cycles for group C, D and E are marked in Fig. 10.

After processing the first 4×4 sub block, the DONE process is activated to execute two operations: calculating the final rate of current CB or TB by fractional rates of five syntax groups in a fractional rate accumulator, and saving newly derived context models in local register arrays. These new context models will be further used to update the 4-level global context models after the mode decision. When a rate estimator reaches the DONE status, it can be scheduled to process another CB or TB.

3.3.7 RD Mode and Size Decision

Mode and size decision in intra prediction involves the determination of best prediction modes and partition for a given CTU. Fig. 3.11 illustrates a partition and mode decision flow, where each RD cost accumulator calculates the sum of 4 sub-CU RD costs and then compares it with the RD cost of a larger CU. RD cost of the same CU with different PUs are also compared, even though this process is not presented in Fig. 3.11. Finally, the smallest RD cost among all possible prediction modes and partitions is taken as the RD cost of this CU, which will be further compared with a larger CU until CU 64×64 is reached. The progressive comparison from the smallest CU to the largest CU guarantees to find the best CTU partition as well as the best prediction mode of each PU in the partition.

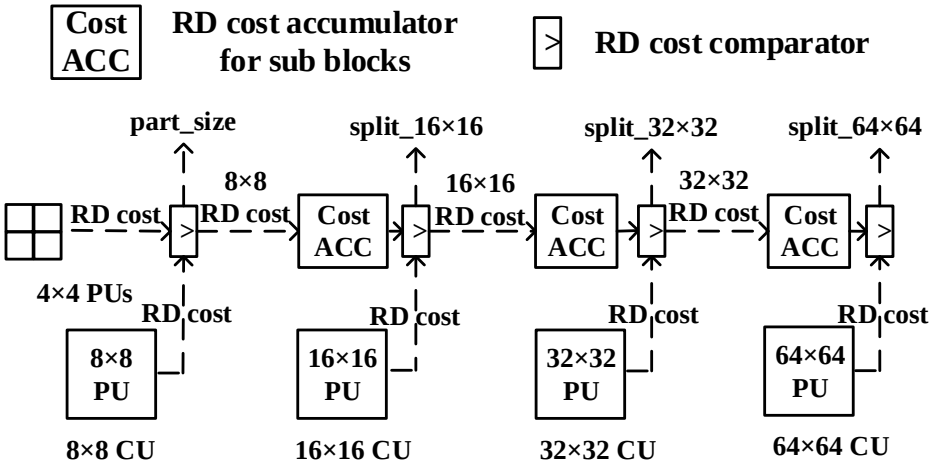


Fig. 3.11. Partition and mode decision flow through RD cost comparison.

Moreover, the RD cost comparison in Fig. 3.11 also generates signals for the context model updating, such as *"part_size"*, *"split_16×16"*, *"split_32×32"*, and *"split_64×64"*. *"part_size"* indicates the partition mode of 8×8 CUs. If *"part_size"* is equal to 1, it means 4×4 PUs have a smaller RD cost than 8×8 PUs, thus partition of 4×4 PUs is chosen. *"split_16×16"* indicates if a 16×16 CU should be split into four 8×8 CUs. "*split_32×32*" indicates if a 32×32 CU should be split into four 16×16 CUs. "*split_64×64*" indicates if a 64×64 CU should be split into four 32×32 CUs.

3.3.8 Context Model Updating

The proposed global context model consists of four levels. Level 0, 1, 2 and 3 are dedicated for 8×8, 16×16, 32×32, and 64×64 CUs, respectively. All global context models are stored in four register arrays, instead of on-chip SRAMs. Each register array saves the context models of a specific CU size. Among the 16 syntax elements in Table 3.2, six of them contain bypass coded bins, and ten contain regular coded bins and hence rely on context models. As listed in Table 3.2, 84 context models are involved in rate estimation and each context model is stored in a 7-bit register. Therefore, there are a total of 84×4 7-bit registers in the table for 4-level

context models.

According to the input signal *"CU_size"*, context models for a specific CU size are loaded into the local register arrays for syntax groups *A-D*. As been introduced in section 3.3.1, the use of local context models reduces data access time and avoids improper update of the global context models. For an 8×8 CU, its partition modes (*i.e.*, PART_2*N*×2*N* and PART_*N*×*N*) share the same initial values of context models. The global context models update begins after the rate-distortion mode decision of a certain CU block. *"CU_size"* specifies which level of global context models to update. If *"CU_size"* indicates it is a 16×16 CU, the corresponding global context models belong to level-1. After comparing the RD costs of this 16×16 CU and its four 8×8 sub-CUs, the new context models will be selected by mode decision to update both level-1 and level-0 of the global context models.

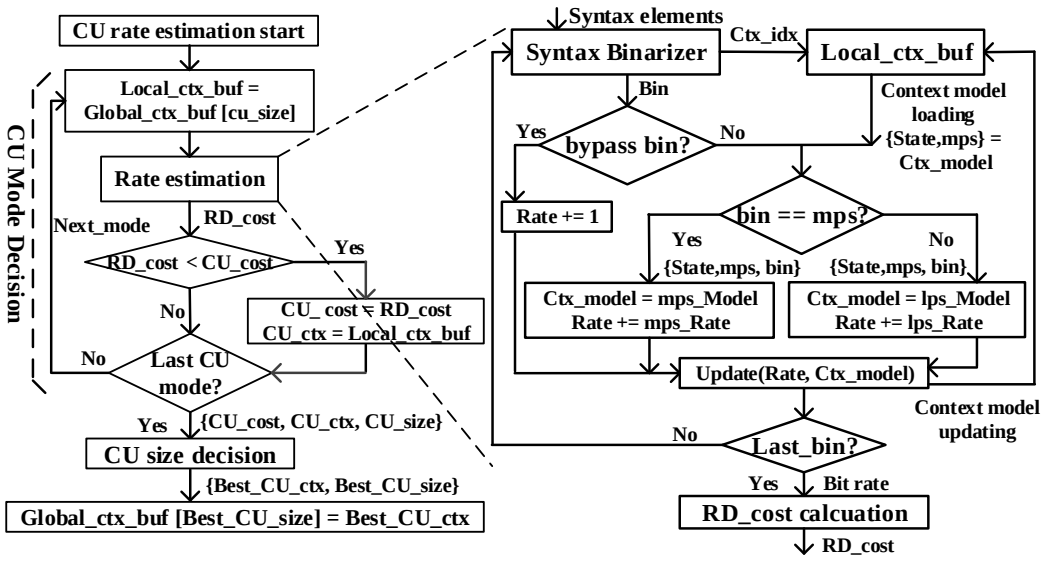


Fig. 3.12. Flow chart of context model loading and updating scheme for each CU.

The flow chart of context model loading and updating scheme of each CU is illustrated in Fig. 3.12. For each CU, all of its prediction modes share the same set of initial context models, which are firstly loaded from the global context model buffer into the local context model buffer

at the beginning of the rate estimation process for each prediction mode. Then, the rate estimation process of one CB starts, syntax binarizer first converts syntax elements into bins. For a bypass bin, the total rate will increase by 1. For a regular bin, according to its context model and bin value, its rate is estimated through look-up tables. Specifically, if a regular bin is equal to the most probable symbol (*mps*), the total rate will increase by *mps_Rate* and the new context model is derived from *mps_Model*. Otherwise, if a regular bin is equal to the least probable symbol (*lps*), the total rate will increase by *lps_Rate* and the new context model is derived from *lps_Model*. After the rate estimation for each bin, the new context model replaces the current context model in the local context buffer. Then, after the rate estimation for all bins, the bit rate and *RD_cost* for each prediction mode are derived and calculated. In this way, each CU prediction mode corresponds to a particular set of updated local context models and an associated *RD_cost*. Through comparing these RD costs among all prediction modes, the best CU prediction mode which corresponds to the smallest *RD_cost* is found. The smallest *RD_cost* and associated local context models are saved as "*CU_cost*" and "*CU_ctx*", respectively. Next, after CU size decision, the best CU size and the best CU context models are determined. These results will be used to update the global context models.

3.4 IMPLEMENTATION RESULTS

Experiments have been conducted to evaluate the efficiency of the proposed rate estimator. A benchmark containing only luma rate estimator has been established, while the chroma rate is estimated by non-zero quantized coefficients. A strategy of mode reduction is first applied to all PUs. For luma 4×4 PUs, three most probable modes, one regular mode (planar, DC, vertical, horizontal), and three extra modes with minimal Hadamard cost are evaluated by RDO. For luma 8×8, 16×16, and 32×32 PUs, three most probable modes and one extra mode

with minimal Hadamard cost are sent to the RDO process. For luma 64×64 PUs, only the most probable mode is checked by the RDO process. The best luma prediction mode will be selected according to the RDO cost which is calculated using distortion and rate. Here rate is estimated by the CABAC rate estimation algorithm. For all chroma PUs, four regular prediction modes, including planar, DC, vertical, and horizontal modes, are supported. Chroma mode is selected by a modified RDO algorithm, in which chroma rate is simply replaced by the number of non-zero coefficients after quantization. Compared with HM 15.0, the benchmark introduces about 4.25% BD-Rate and 0.24 dB BD-PSNR losses for 21 video test sequences with QP values (22, 27, 32 and 37). The proposed rate estimator is then applied to this benchmark and brings a decrease of 0.005% in BD-Rate and an increase of 0.0092dB in BD-PSNR gain against the benchmark. The comparison results have been illustrated in Table 3.4. Thus, the luma-based CABAC rate estimation is verified to be a reasonable approach that retains superior compression efficiency and low complexity.

The proposed hardware architecture has been implemented in Verilog and synthesized in FPGAs and ASICs. Since the proposed hardware architecture is identical to the table-based CABAC bit rate estimation algorithm in HM 15.0 reference software except ignoring the flag *"split_cu_flag"*, we first study the impact of ignoring this flag to rate estimation. A proposed hardware design of luma-based rate estimator without *"split_cu_flag"* has been implemented and compared with the luma-based CABAC rate estimation in HM 15.0. As shown in the last two columns in Table 3.4, for all video test sequences, our proposed hardware architecture decreases the BD-Rate by 0.005% and increases the BD-PSNR by 0.0092dB. Therefore, the proposed hardware architecture improves the compression efficiency than the table-based luma-only CABAC rate estimation in HM. Omitting "*split_cu_flag*" in the proposed hardware

architecture improves the video compression efficiency and also results in lower hardware complexity.

Table 3.4 Comparison of experimental results between the original rate estimation algorithm in HM and the proposed modified rate estimation algorithm

Class	**Sequences**	**Rate Estimation in HM (luma-only) vs. Rate Estimation in Standard HM**		**Proposed Rate Estimator (luma-only, without *"split_cu_flag"*) vs. Rate Estimation in HM (luma-only)**	
		BD-Rate[%]	BD-PSNR[dB]	ΔBD-Rate[%]	ΔBD-PSNR[dB]
A	People On Street	3.4417	-0.1680	-0.1935	0.0093
	Traffic	4.7199	-0.2160	-0.0913	0.0039
B	Park Scene	3.5296	-0.1358	-0.1252	0.0047
	Kimono	3.0720	-0.0983	0.6355	-0.0221
	Basketball Drive	3.6115	-0.0954	0.2915	-0.0079
	BQ Terrace	2.4140	-0.0935	-0.0633	0.0023
	Cactus	4.1528	-0.1300	-0.0126	0.0000
C	Basketball Drill	8.6861	-0.3730	-0.2156	0.0088
	BQ Mall	3.3237	-0.1625	-0.1520	0.0075
	Party Scene	3.4397	-0.2052	-0.0890	0.0051
	Race Horses C	4.0470	-0.1967	-0.0425	0.0022
D	Basketball Pass	4.7913	-0.2554	-0.0645	0.0034
	Blowing Bubbles	3.5473	-0.2011	-0.0740	0.0042
	BQ Square	3.0824	-0.1945	-0.0444	0.0026
	Race Horses	5.1280	-0.4363	-0.1001	0.1752
E	Kristen And Sara	4.1179	-0.1883	0.2092	-0.0096
	Four People	3.2739	-0.1698	-0.0528	0.0024
	Johnny	2.6748	-0.1067	0.1167	-0.0045
F	Slide Editing	5.1133	-0.6313	-0.0548	0.0072
	Slide Show	7.0305	-0.5850	0.0217	-0.0010
	China Speed	6.0791	-0.4702	-0.0034	0.0000
Average value		**4.25**	**-0.24**	**-0.005**	**0.0092**

For various sizes (4×4 PUs, 8×8 CUs, 16×16 CUs, 32×32 CUs, and 64×64 CUs) and QP values (22, 24, 26, 28, 30, 32, 34 and 37), Table 3.5 shows the obtained PSNR and the required number of clock cycles to accomplish rate estimation using the proposed rate estimator. Experimental results of all test sequences in class A and B are provided. In time-constrained, high-performance video coding applications, this table is useful for fast CU decision in real-time operation. For example, based on the PSNR expectation and the maximum allowable time to process, the best CU size can be roughly determined based on Table 3.5. For another example, if

QP 22 is selected for sequence “People on Street”, if one 16×16 CU and four sub 8×8 CUs result in the same RD cost, the 16×16 CU needs 145 clock cycles for rate estimation, while four sub 8×8 CUs need 162 clock cycles for rate estimation. From a processing time point of view, this 16×16 CU is advantageous since 17 clock cycles are saved.

Table 3.5 Experimental results of PSNR and number of clock cycles for different PU/CU sizes and QP values of our proposed design

		QP = 22						QP = 24					
Class	Sequence	PSNR	4×4	8×8	16×16	32×32	64×64	PSNR	4×4	8×8	16×16	32×32	64×64
A	People on Street	43.28	11.3	40.4	144.9	560	2232	42.40	10.8	35.6	124.0	480	1912
A	Traffic	43.47	11.0	37.4	131.1	512	2043	42.65	10.4	33.4	111.3	436	1740
B	Basketball Drive	43.28	11.8	53.0	184.6	646	2574	42.40	10.2	37.5	123.1	462	1851
B	BQ Terrace	43.65	14.1	63.6	240.0	937	3756	42.90	13.1	58.3	217.3	850	3401
B	Cactus	42.92	13.5	62.9	244.1	931	3720	41.91	11.8	49.7	190.6	760	3035
B	Kimono	44.00	10.0	27.6	89.6	336	1313	43.36	9.4	22.0	58.3	232	904
B	Park Scene	42.65	12.0	48.4	182.6	722	2892	41.54	11.0	40.7	145.1	594	2380
		QP = 26						QP = 28					
Class	Sequence	PSNR	4×4	8×8	16×16	32×32	64×64	PSNR	4×4	8×8	16×16	32×32	64×64
A	People on Street	41.61	10.2	31.0	105.0	405	1611	40.93	9.7	27.3	88.4	341	1357
A	Traffic	41.84	9.9	29.7	94.4	365	1457	41.08	9.4	26.6	79.9	306	1222
B	Basketball Drive	41.66	9.1	27.1	81.7	308	1249	41.05	8.5	22.4	60.5	220	901
B	BQ Terrace	42.13	11.9	50.8	188.2	739	2952	41.39	10.9	43.2	157.1	627	2500
B	Cactus	40.90	10.5	37.6	136.2	579	2316	39.98	9.7	30.8	100.1	416	1664
B	Kimono	42.69	8.9	19.6	43.5	165	650	41.99	8.6	18.4	36.0	127	506
B	Park Scene	40.40	10.3	35.2	118.8	481	1929	39.37	9.8	30.9	99.0	393	1577
		QP = 30						QP = 32					
Class	Sequence	PSNR	4×4	8×8	16×16	32×32	64×64	PSNR	4×4	8×8	16×16	32×32	64×64
A	People on Street	40.35	9.2	24.5	75.3	288	1147	39.63	8.8	22.0	62.8	242	964
A	Traffic	40.44	9.0	24.0	68.5	258	1030	39.63	8.7	21.7	57.8	217	866
B	Basketball Drive	40.53	8.1	19.8	48.9	175	719	39.89	7.8	17.9	40.1	144	595
B	BQ Terrace	40.77	10.2	36.9	131.8	527	2102	39.97	9.6	31.9	109.5	442	1765
B	Cactus	39.19	9.2	26.6	80.8	320	1287	38.27	8.8	23.4	66.0	257	1035
B	Kimono	41.41	8.3	17.7	31.2	103	412	40.63	8.1	17.0	27.2	86.1	347
B	Park Scene	38.46	9.3	27.2	83.2	322	1290	37.45	8.9	23.7	68.1	260	1039
		QP = 34						QP = 37					
Class	Sequence	PSNR	4×4	8×8	16×16	32×32	64×64	PSNR	4×4	8×8	16×16	32×32	64×64
A	People on Street	38.90	8.5	20.1	52.6	201	804	37.98	8.1	17.9	40.8	153	618
A	Traffic	38.80	8.4	19.9	49.2	180	723	37.76	8.1	17.7	38.6	137	551
B	Basketball Drive	39.22	7.7	16.6	34.1	120	496	38.39	7.5	15.4	27.2	92	381
B	BQ Terrace	39.14	9.1	28.2	92.2	370	1474	38.07	8.6	23.8	71.8	280	1119
B	Cactus	37.37	8.4	21.1	55.1	208	842	36.26	8.1	18.4	42.2	153	621
B	Kimono	39.79	7.9	16.4	24.2	73.7	298	38.69	7.7	15.5	20.8	59.5	241
B	Park Scene	36.46	8.5	20.9	55.7	206	825	35.28	8.1	17.9	40.7	145	578

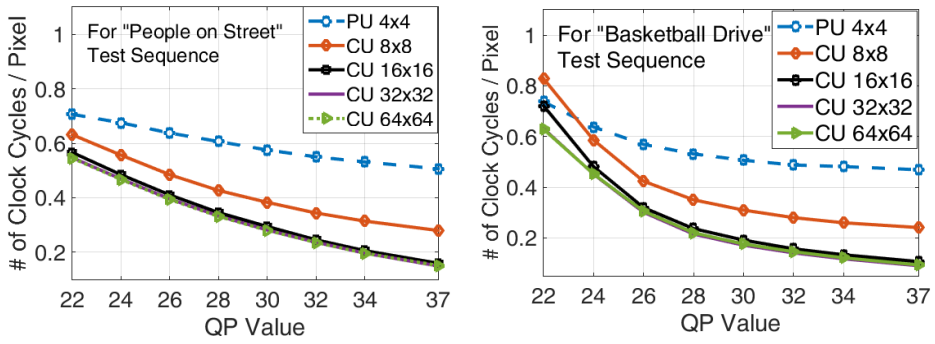


Fig. 3.13. The required number of clock cycles varying with QP values for two test sequences

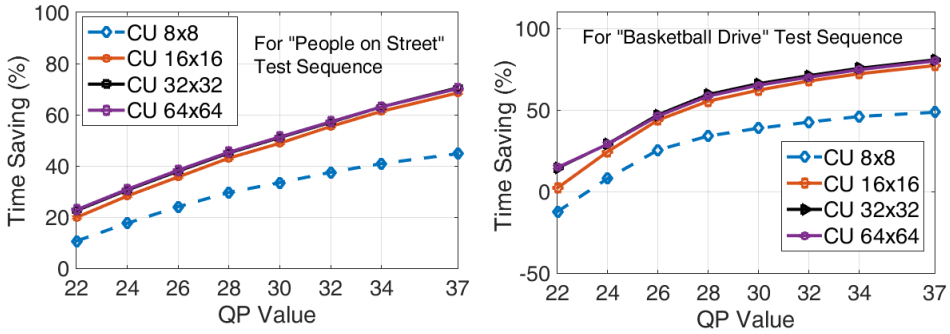


Fig. 3.14. Processing time-saving percentage (with respect to PU 4×4) varying with QP values for two test sequences

For each CU size, Fig. 3.13 plots how the required number of clock cycles per pixel (*i.e.*, the hardware processing time for rate estimation per pixel) varies with QP values. Regardless of CU size, the required number of clock cycles per pixel drops drastically with the increase of QP value for both video test sequences. This is because a larger QP value leads to less non-zero coefficients after quantization step. Thus, fewer syntax elements go through the binarization process. In addition, updating context models is also less frequent. Therefore, the required processing time per pixel for rate estimation is reduced as QP increases.

Fig. 3.14 plots time-saving percentages of CUs from 8×8 to 64×64 with respect to PUs 4×4 for different QP values. Despite all CU sizes with larger QP values require less processing time of rate estimation, it is observed that the curves for CUs 16×16, 32×32 and 64×64 are

almost identical with the most notable time-saving. Therefore, for a given QP value, the throughput of our proposed rate estimator is the highest for CUs 16×16, 32×32 and 64×64. This observation is explained as follows. As we know, the bit rate is composed of coded bits of all quantized coefficients and header syntax. Quantized coefficients scale down with a larger QP value, while header syntax does not. With a larger quantization step in high QP scenarios, a smaller number of coded bits of quantized coefficients is obtained. Thus, header syntax gradually dominates the processing time of bit rate estimation for larger CUs.

Table 3.6 Resource consumption comparison of rate estimation hardware designs.

Architecture	Pastuszak [29]	This work (for a proposed rate estimator instance)	
FPGA Platform	Arria II GX [ALUT]	Arria II GX [ALUT + Registers]	Altera Stratix V GX [ALUT + Registers]
Logic Gate Count	8441 (for main reconstruction loop) 3909 (for 4×4 reconstruction loop)	3779 + 1304 *	3764 + 1262 **
Frequency (MHz)	100 (for main reconstruction loop) 200 (for 4×4 reconstruction loop)	115	208
Supported PU/CU sizes	8×8, 16×16, 32×32 CUs	4×4 PU, 8×8, 16×16, 32×32, 64×64 CUs	

*197 ALUTs and 442 registers are used for global context models and initialization logic, which are shared by multiple rate estimator instances
**166 ALUTs and 393 registers are used for global context models and initialization logic, which are shared by multiple rate estimation instances

The proposed hardware architecture of the rate estimator is synthesized for Arria II GX and Altera Stratix V GX FPGA platforms. The resource consumption, frequency, rate estimation performance and supported TU/CU sizes are provided in Table 3.6. The proposed highly-parallel rate estimation architecture has also been synthesized and implemented in an ASIC design using TSMC 90nm technology as shown in Table 3.7. The results of our proposed design are obtained from the Synopsys Design Compiler under the normal corner of TSMC 90nm process, at a 1.0V supply voltage and room temperature. There are a total of 13 rate estimators being instantiated in order to satisfy the throughput requirement of the system (3840×2160p @ 30fps). The numbers of rate estimators corresponding to 4×4, 8×8, 16×16, 32×32, and 64×64 PUs are 7, 2, 2, 1, and 1 respectively. An HEVC intra encoder hardware is also implemented to evaluate the entire intra

encoder throughput using our proposed rate estimators. As shown in Table 3.7, the entire intra encoder system sustains the real-time encoding of 3840×2160p @ 30fps.

Table 3.7 Hardware design comparison of rate estimators

Hardware Architecture	[23]	[28]	[29]	[32]	[33]	This work
Rate Estimation Algorithm	Magnitude of non-zero coefficients	Binary classification of *N*×*N* quantized coefficients	Bin counting	CFBAC	Parallelized CABAC context adaption	Highly-parallel CABAC without the syntax "*split_cu_flag*"
Rate model preprocessing	Required	Required	Required	No need	No need	No need
Video Test Sequences	8 in class A-C	21 in Class A-E	24 in Class A-E, 4K	N/A	Class A-F	21 in Class A-E
Technology	N/A	TSMC 90nm	TSMC 90nm	TSMC 28nm	SMIC 55nm	TSMC 90nm
Area (k gate)	N/A	N/A	53.1	56.8 (for binarization) + 120.4 for (CFBAC Rate estimator)	N/A	197 (for 13 rate estimator instances)
Power (mW)	N/A	N/A	13.3	N/A	N/A	76
Frequency (MHz)	N/A	357	200 (for main reconstruction loop) 400 (for 4×4 reconstruction loop)	312	294	320
Supported CU/PU sizes	N/A	4×4 ~ 32×32 PU 8×8 ~ 32×32 CU	8×8 ~32×32 CU	16×16 ~ 64×64 CU	4×4 ~ 32×32 PU 8×8 ~ 32×32 CU	4×4 ~ 64×64 PU 8×8 ~ 64×64 CU
Relationship among clock cycles, QP, CU size, PSNR	No	No	No	No	No	Yes, in table V
Rate estimation performance	Low BD-Rate [6.27%] BD-PSNR [-0.26dB]	Low BD-Rate [4.53%] BD-PSNR [-0.20dB]	Low BD-Rate [2.11%] BD-PSNR [-0.091dB]	Medium BD-Rate [1.1%] BD-PSNR [N/A dB]	Medium BD-Rate [0.5%] BD-PSNR [N/A dB]	High BD-Rate [-0.005%] BD-PSNR [0.0092dB]
Encoder throughput with the specific rate estimator	N/A	1920×1080 @44fps	3840×2160 @30fps	8192×3420 @30fps	1920×1080 @60fps	3840×2160 @30fps

Table 3.7 summarizes the comparison of this work with existing rate estimation hardware designs in the literature. The algorithm for rate estimation, implementation technology, area, power, frequency, supported TU sizes, the relationship among clock cycles, QP, CU size and

PSNR, and rate estimation accuracy are included to demonstrate the benefits of our proposed design. The non-zero coefficient counting algorithm in [23, 28] and the bin counting algorithm in [29] result in lower accuracy, because they are incompatible with the default CABAC-based rate estimation approach. These works [23, 28, and 29] require preprocessing to establish simplified rate estimation models. Due to these simplified rate models are experimentally determined from limited video test sequences, there is no rigid theoretical proof to bind the rate estimation accuracy in general video scenarios. Hence, the accuracy of simplified rate estimation model may vary widely depending on video contents. The CFBAC algorithm in [32] utilizes fixed context models without adaptive updates. No design effort has been reported in [32] to improve the level of parallelism for rate estimation. The rate estimator hardware costs 56.8k gates for binarization and 120.4k gates for CFBAC rate estimation. Compare to our hardware cost (*i.e.*, 197k gates), our proposed design results in an increase of 11% in hardware resources. Meanwhile, our proposed design leads to a decrease of 0.005% in BD-Rate, while the work [32] obtains an increase of 1.1%. The CABAC architecture in [33] improves the throughput of CABAC-based rate estimator by context adaption of 2 bottleneck syntax elements, while the remaining 14 syntax elements are still processed in a serial manner. The required hardware cost and power consumption in [33] are not reported. Besides, 64×64 CUs are not supported in [33], while it is included in our proposed rate estimator. In contrast with the reference [29], under an iso-throughput condition (i.e., 3840×2160 @30fps), even though the hardware area and power consumption of our rate estimator instances are worse. As we have discussed earlier, the design in [29] requires rate model preprocessing, since the parameters in the rate estimation model need to be determined through statistical data collection and curve fitting. Such a data-driven statistical rate estimation model cannot guarantee good accuracy of bit rate estimation for any

given video file. In contrast, the proposed CABAC rate estimator fully conforms to the computational theory and procedure of CABAC bit estimation (*i.e.,* binarization and context-adaptive probability modeling), so good rate estimation accuracy can be ensured and the estimation accuracy may change very slightly with video contents. Therefore, the rate estimation accuracy and reliability of our proposed design are much improved.

3.5 CONCLUSION

Despite the great bit rate estimation performance, hardware implementations of CABAC bit rate estimator have been impeded for a long time due to its low throughput. To deal with this challenge, we propose a highly-parallel hardware architecture of the CABAC rate estimator in this chapter. Details of rate estimation algorithm, methodology, circuit diagram, and hardware implementation results are described and discussed. Compared with existing related works in the literature, this proposed architecture demonstrates significant advantages in rate estimation accuracy and reliability, with the overhead of a relatively larger chip area and higher power consumption. This proposed design supports resolutions up to 3840×2160 @30fps with a decrease of 0.005% in BD-Rate. To our best knowledge, this is the first study that reports a high-throughput CABAC rate estimator with the best rate estimation performance.

CHAPTER 4

PROPOSED HEVC INTRA ENCODER

In this chapter, we propose efficient algorithm adaptations and fully-parallel hardware architecture of H.265/HEVC intra encoder, which is capable of sustaining real-time compression of videos at 4K@30fps. The proposed algorithm adaptations are hardware-oriented and aim at dramatically reducing the coding complexity of H.265/HEVC intra coding. Fully-parallel hardware architecture of H.265/HEVC intra encoder that realizes parallel processing of four different PU sizes simultaneously is proposed. To our best knowledge, this is the first work to use 4-parallelism in intra prediction. Along with complexity reduction algorithms, the proposed intra encoder achieves a 27% workload reduction with 4.39% and 0.21dB coding performance degradation on average in BD-Rate and BD-PSNR, respectively. Hardware implementation in FPGA and ASIC shows our design can run at 120MHz and 320MHz, respectively. Compared with the state-of-the-art designs, our proposed design demonstrates advantages in computational complexity, bit rate, video quality, throughput, reliability, and flexibility.

4.1 INTRODUCTION

With the growing demand for superior video compression performance for UHD video applications, HEVC was proposed to replace its predecessor H.264. HEVC standard was devised to reduce more than 50% bit rate at the same level of video quality. However, the increased complexity and data/timing dependency has drastically impeded its throughput in hardware implementation. It is certain that the new features (*e.g.*, coding tree unit, quad-tree structure, extended prediction directions, expanded block sizes, etc.) in HEVC bring about excellent coding performance compared with previous standards. Despite the research efforts made in software algorithms and hardware architectures, so far, the optimization of HEVC intra encoders

has not been widely explored, especially in hardware friendly algorithm adaptations.

As introduced in the previous chapter, there are two major bottlenecks that impede the throughput of HEVC intra encoders. The first one is the increased computation. The new features of HEVC make the rate-distortion optimization process extremely complex and require lots of computation [34]. For example, in the full search scheme, each CTU partition has to be estimated to find the best partition, and each PU has to be traversed 35 times to find the best prediction mode. However, the original RDO process consists of a series of processes such as prediction, transform, quantization, inverse quantization, inverse transform, reconstruction, rate estimation, and distortion calculation. The RDO process is not only computation-intensive but also time-consuming [35]. Moreover, the strict time and throughput requirements for practical video encoders make it infeasible to perform exhaustive mode and partition searching [36-38]. Therefore, it is indispensable to develop efficient algorithms to reduce HEVC's computational complexity, while retaining excellent video compression efficiency. The second bottleneck is the strong data/timing dependency during the RDO process. For example, intra prediction of each block relies on reference pixels, which are derived from reconstructed pixels of neighboring blocks. This data dependency between coding blocks leads to low-throughput serial processing of coding blocks. The low throughput CABAC based rate estimation is also caused by context model dependency between syntax elements. In addition, the CU size mode decision depends on not only the RD cost of current CU but also RD cost of sub CUs. All those dependencies have significantly affected HEVC's throughput [39-45]. Thus, it is worth investigating on how to relieve the data/timing dependency in various computational tasks.

In this chapter, we propose a new HEVC intra encoder that computes the full cycle of intra prediction, transform, quantization, inverse quantization, inverse transform, reconstruction,

and rate estimation in a fully parallel manner. The proposed intra encoder consists of two parts: efficient HEVC algorithm adaptations and highly-parallel hardware architecture design. The former aims to reduce the computational complexity in the algorithm level, while the latter maximizes the potential of parallelism to improve the overall throughput of intra encoder. The proposed intra encoder supports all CU/PU/TU sizes and 35 prediction modes. Compared with HM-15.0, the proposed algorithm adaptations lead to a 27% computation reduction with an average loss in BD-Rate and BD-PSNR is 4.39% and 0.21dB, respectively. To address the bottleneck of data/timing dependency, a fully-parallel intra encoder architecture utilizing 4-parallelism in intra prediction is proposed. Intra prediction of four different size PUs from 4×4 to 32×32 will be performed simultaneously in 4 prediction engines (PE) to greatly improve prediction throughput. Highly pipelined computational schemes are designed and employed in each PE to maximize RDO throughput. Moreover, the proposed high throughput table-based CABAC rate estimator in chapter 3 is incorporated in the proposed intra encoder to further increase RDO performance. Experimental results show the proposed intra encoder is capable of handling real-time video compression for 4K videos at 30fps.

The rest of this chapter is organized as follows. The proposed efficient algorithm adaptations are described in section 4.2. Section 4.3 describes the hardware architecture and timing diagrams. In section 4.4, system implementation and results are presented and compared with the state-of-the-art designs in the literature. Finally, section 4.5 concludes this chapter.

4.2 PROPOSED EFFICIENT ALGORITHM ADAPTATIONS

To reduce the computational complexity of HEVC intra encoders and make a better tradeoff between complexity and performance, we propose four hardware-oriented algorithm adaptations for HEVC intra prediction. All proposed algorithm adaptations have been validated

in HM-15.0 reference software and comparison has been made to evaluate compression performance. Compared with HM-15.0, our approach achieves a workload reduction of 27% on average with a 4.39% BD-Rate increase and 0.21dB BD-PSNR decrease [47]. The resulting compression efficiency of proposed algorithms has been illustrated in Table 4.1. Details of each algorithm adaptation are elaborated and discussed in the following sub-sections.

Table 4.1 Losses in compression efficiency for successive modifications: PU chroma mode preselection (M1), PU luma mode preselection (M2), Modified CU mode decision (M3), simplified CABAC rate estimator (M4).

		M1		M1+M2		M1+M2+M3		M1+M2+M3+M4	
Class	**Sequences**	**BD-Rate [%]**	**BD-PSNR [dB]**	**BD-Rate [%]**	**BD-PSNR [dB]**	**BD-Rate [%]**	**BD-PSNR [dB]**	**BD-Rate [%]**	**BD-PSNR [dB]**
A	People On Street	1.38	-0.07	2.25	-0.12	2.93	-0.15	3.31	-0.17
	Traffic	3.07	-0.15	3.22	-0.16	4.10	-0.20	4.49	-0.22
B	Park Scene	1.21	-0.05	1.05	-0.05	1.71	-0.08	2.04	-0.09
	Kimono	1.71	-0.05	0.89	-0.03	2.10	-0.06	2.92	-0.09
	Basketball Drive	2.82	-0.08	3.00	-0.09	3.96	-0.12	4.33	-0.13
	BQ Terrace	1.07	-0.05	1.49	-0.07	1.86	-0.09	1.95	-0.09
	Cactus	2.36	-0.08	2.90	-0.10	3.63	-0.13	3.87	-0.14
C	Basketball Drill	9.01	-0.44	11.28	-0.55	11.94	-0.58	12.12	-0.59
	BQ Mall	1.88	-0.10	2.74	-0.14	3.13	-0.16	3.30	-0.17
	Party Scene	1.44	-0.11	2.61	-0.20	2.70	-0.20	2.76	-0.21
	Race Horses	3.26	-0.19	4.15	-0.25	4.75	-0.28	4.97	-0.29
D	Basketball Pass	4.17	-0.26	5.42	-0.33	5.83	-0.36	6.02	-0.37
	Blowing Bubbles	1.86	-0.14	3.49	-0.26	3.57	-0.26	3.63	-0.27
	BQ Square	0.32	-0.02	2.17	-0.17	2.34	-0.18	2.37	-0.18
	Race Horses	4.01	-0.27	5.51	-0.36	5.95	-0.39	6.12	-0.40
E	Kristen And Sara	3.90	-0.16	4.69	-0.20	5.80	-0.24	6.28	-0.26
	Four People	1.57	-0.08	2.14	-0.10	2.94	-0.14	3.33	-0.16
	Johnny	3.38	-0.12	3.77	-0.13	4.81	-0.17	5.31	-0.19
F	Slide Editing	1.40	-0.21	2.37	-0.35	2.48	-0.36	2.48	-0.36
	Slide Show	3.85	-0.34	5.90	-0.52	6.66	-0.59	6.72	-0.59
	China Speed	2.10	-0.19	3.82	-0.34	4.06	-0.36	4.12	-0.37
4K	Beauty	0.13	-0.004	0.52	-0.01	4.07	-0.07	5.06	-0.08
	Bosphorus	1.91	-0.05	1.09	-0.03	3.82	-0.10	6.19	-0.16
	Honey Bee	0.56	-0.02	0.93	-0.03	2.18	-0.02	2.31	-0.04
	Jockey	0.52	-0.02	1.05	-0.02	3.69	-0.04	4.54	-0.08
	Ready Steady Go	2.59	-0.06	2.15	-0.05	3.93	-0.09	4.84	-0.11
	Shake & Dry	0.11	-0.004	0.10	-0.005	0.13	-0.006	1.84	-0.04
	YachtRide	3.55	-0.10	3.08	-0.09	4.69	-0.13	5.79	-0.16
Average value		2.33	-0.12	2.99	-0.17	3.92	-0.20	4.39	-0.21

4.2.1 PU Chroma Mode Preselection

As each PU contains one luma and two associated chroma blocks, specific mode preselection schemes are proposed to reduce the computational complexity for luma and chroma

blocks, respectively. Fewer preselected prediction modes usually result in a less complex RDO process but more degradation in coding efficiency. Considering the tradeoff between coding efficiency and computational complexity, an appropriate set of prediction modes is assigned to each PU. As it takes more clock cycles to process larger PUs, fewer modes are selected for larger PUs to meet the same timing budget in different PEs.

In the standard algorithm of H.265/HEVC, the RDO process of PU chroma blocks contains five prediction modes, including one derived mode from the luma RDO process and four regular modes (*i.e.*, DC, Planar, Vertical, and Horizontal). If the derived luma mode happens to be within the four regular modes, the angular mode 34 should be used. Efficient hardware implementation of prediction and transformation is easily designed for these known regular modes. However, for the unknown derived mode, chroma prediction cannot start until the completion of the luma RDO process. This data & timing dependency is an obstacle to efficient hardware implementation in chroma mode prediction. In order to eliminate this data dependency, our chroma mode preselection scheme excludes this derived luma mode. Thus, the best chroma mode is selected only from the four regular modes. Moreover, in order to further reduce computational complexity, the chroma rate is estimated based on the number of non-zero coefficients after quantization stage [21]. This algorithm adaptation enables a fast RDO process for chroma blocks. According to the experimental results in Table 4.1, the algorithm adaptation for chroma mode reduction causes a video quality loss (an average BD-PSNR of 0.12dB) and a bit rate increase (an average BD-Rate of 2.33%).

4.2.2 PU Luma Mode Preselection

In the proposed PU luma mode preselection algorithm, most probable modes (MPMs) and Hadamard transform are used for low cost and fast computation. For 64×64 PUs, only the

first MPM defined by the HEVC standard is selected for RDO, so no mode decision is required. However, it is still needed to calculate the RD cost of 64×64 PUs, since the cost is useful for later CU partition decisions. For 32×32 PUs, in addition to three MPMs, one mode that corresponds to the least Hadamard cost out of 35 prediction modes is selected. For 16×16 and 8×8 PUs, the RDO process involves three MPMs, one regular mode, and one Hadamard mode that is determined by the minimum Hadamard cost. No common mode is shared among the MPMs, regular mode, and Hadamard mode. Because 4×4 PUs correspond to the largest number of blocks in a CTU, RDO processing of 4×4 PUs is the bottleneck for timing and hardware implementation. Moreover, 4×4 PUs are indispensable because of their excellent compression performance for shaper detail-rich video sequences. Thus, it is critical to pre-select appropriate prediction modes for 4×4 PUs. In the proposed algorithm, seven prediction modes are selected as candidate modes for 4×4 PUs, including MPMs, angular modes, and one mode selected from the four regular modes. These angular modes are selected from 12 angular modes of 2, 4, 6, 8, 11, 15, 19, 23, 28, 30, 32, and 34 based on Hadamard cost. Experimental results in Table 4.1 show that the PU chroma and luma mode preselection leads to an average increase of 2.99% in BD-Rate and an average decrease of 0.17dB in BD-PSNR.

4.2.3 Modified CU Mode Decision

According to the quad-tree structure in H.265/HEVC, each CU is allowed to be split into four smaller CUs until reaching the minimum CU size. In order to find the best CU partition, CU mode decision is performed. In the HM-15.0 software, the CU-level rate estimation does not directly use the sum of luma and chroma rates computed in the prior luma and chroma RDO processes. Instead, the pre-selected best luma and chroma modes are used in the HM-15.0 to calculate the CU-level rate, followed by the CU-level RD cost calculation in the CU mode

decision. This CU-level rate estimation is time-consuming, and it is difficult to implement in hardware architectures. To address this challenge, we propose a low-complexity coding unit mode decision (CUMD) scheme, which computes the RD costs of CU and sub-CU blocks and then determines the best CU partition. Instead of recalculating RD costs of CUs, our CUMD scheme reuses the derived RD costs from prior luma and chroma RDO processes. This algorithm adaptation reduces computational workload and hardware complexity. Table 4.1 shows that the algorithm adaptation for modified CU mode decision causes an increase of 0.93% in BD-Rate and a decrease of 0.03dB in BD-PSNR.

4.2.4 Simplified CABAC Rate Estimation

In the HM-15.0 software, the CU rate is estimated by utilizing the look-up-table (LUT) based CABAC algorithm, which is hardware-friendly and easy to implement. However, because of its strong data dependency in serial syntax processing, the low throughput of CABAC rate estimation prevents it from hardware implementation in UHD video coding systems. Instead, due to less data dependency and high throughput, approximate rate estimation algorithms (*e.g.*, bin counting [29]) are proposed as alternative approaches. Yet, these alternative rate estimation algorithms are based on empirical correlations between bit rate and other parameters (*e.g.*, the number of bins in [29]). As been discussed in [29], these estimated correlations vary with quantization parameter (QP) and video sequences. As a result, with a temporal and spatial variability of QP and video sequences, these empirical rate estimation algorithms cannot guarantee accurate and reliable performance. In contrast, the CABAC based rate estimation algorithm conducts the same binarization and context-adaptive probability modeling as CABAC bitstream encoder, thus it theoretically ensures good accuracy and reliability. Here, we focus on efficient algorithm adaptations for high-throughput CABAC rate estimation.

In this work, three hardware-oriented algorithm adaptations are proposed to improve throughput in CABAC based rate estimation. The first adaptation is to round the floating-point parameter λ, which plays an important role in the Lagrangian equation to tradeoff video quality and compression ratio. In order to avoid hardware-unfriendly arithmetic calculations, a look-up table is established to represent the correspondence between the integer values of λ and QP. The second algorithm adaptation takes place in luma mode rate estimation. A syntax element "*split_cu_flag*" is proposed to be omitted from luma rate estimation, due to the following considerations. From an algorithm point of view, this flag only involves in the comparison between CU and sub-CUs. This syntax element is not involved in rate-distortion comparison among various prediction modes of a given CU size. Removal of this syntax element from the rate estimation procedure reduces computational complexity and simplifies hardware implementation. The third algorithm adaptation reduces the computational workload of CABAC rate estimation. For example, in the HM-15.0 reference software, rate estimation is executed twice for four 4×4 PUs in an 8×8 CU. The first time occurs in intra prediction, where data dependency of CABAC context models is ignored and the same initial context models are used for each 4×4 PU. Then, after the best prediction modes are selected for four 4×4 PUs, the total rate of four 4×4 PUs within this 8×8 CU is estimated again. At this same, data dependency of context models for four 4×4 PUs is taken into account. In this default manner, repeated rate estimation is very time consuming and hardware unfriendly. In order to reduce computational workload, in our proposed algorithm, the rate estimation of four 4×4 PUs is calculated only once, and data dependency of CABAC context models is considered. The experimental results in Table 4.1 show that the simplified CABAC based rate estimation algorithm leads to an increase of 0.47% in BD-Rate and a decrease of 0.01dB in BD-PSNR.

4.3 PROPOSED HARDWARE ARCHITECTURE AND TIMING DIAGRAM

Fig. 4.1 shows the hardware architecture overview of proposed fully-parallel H.265/HEVC intra encoder, which supports 8×8 ~ 64×64 CUs, 4×4 ~ 64×64 PUs, 4×4 ~ 32×32 TUs, and two transformations (Discrete Sine Transform (DST) and DCT). DST is dedicated to 4×4 luma blocks, while DCT is adopted in transform coding of remaining TUs. All functional modules in this architecture can be divided into two groups: prediction engine (PE) modules and non-prediction engine (Non-PE) modules. Four parallel PEs (PE_0~PE_3) perform a series of data processing including mode prediction, transformation, quantization, inverse quantization, inverse transformation, reconstruction, CABAC based rate estimation, distortion estimation, and simplified block mode decision (SBMD). Each PE is dedicated to one or two specific PU sizes, as denoted in Fig. 4.1. According to the statistics about the probability proportion of various PU sizes in [36], the percentage for 64×64 and 32×32 PUs is only 1%. Hence, it is cost-effective to share PE_3 for 32×32 and 64×64 PUs. These Non-PE modules perform less computational tasks of reference pixel storage and preparation, neighboring prediction mode storage and fetch, MPM generation, original pixel loading, and CABAC entropy coding.

In this fully-parallel architecture, data dependency among different sizes of PU blocks is mitigated. Therefore, four PEs simultaneously process all sizes of prediction blocks. Moreover, computational tasks inside each PE are pipelined and well scheduled to maximize processing throughput. Although four PEs are implemented, each module is carefully designed to keep low hardware cost. For example, 1D unified DCTs and iDCTs are used for 16×16 and 32×32 TUs to reduce hardware cost for transformation. Due to alleviated data dependency and timing constraints in the proposed hardware architecture, this design is very flexible to realize algorithm-level modifications or extensions, such as change of PU preselection modes, revise of

transformation or quantization blocks, and implementation of new high-throughput rate estimation algorithms. In contrast, the existing hardware architecture in the state-of-the-art design [29] operates in an interleaved processing manner. Due to the inherently strict data dependency and tight timing schedule, it is more challenging to make efforts to modify or extend it.

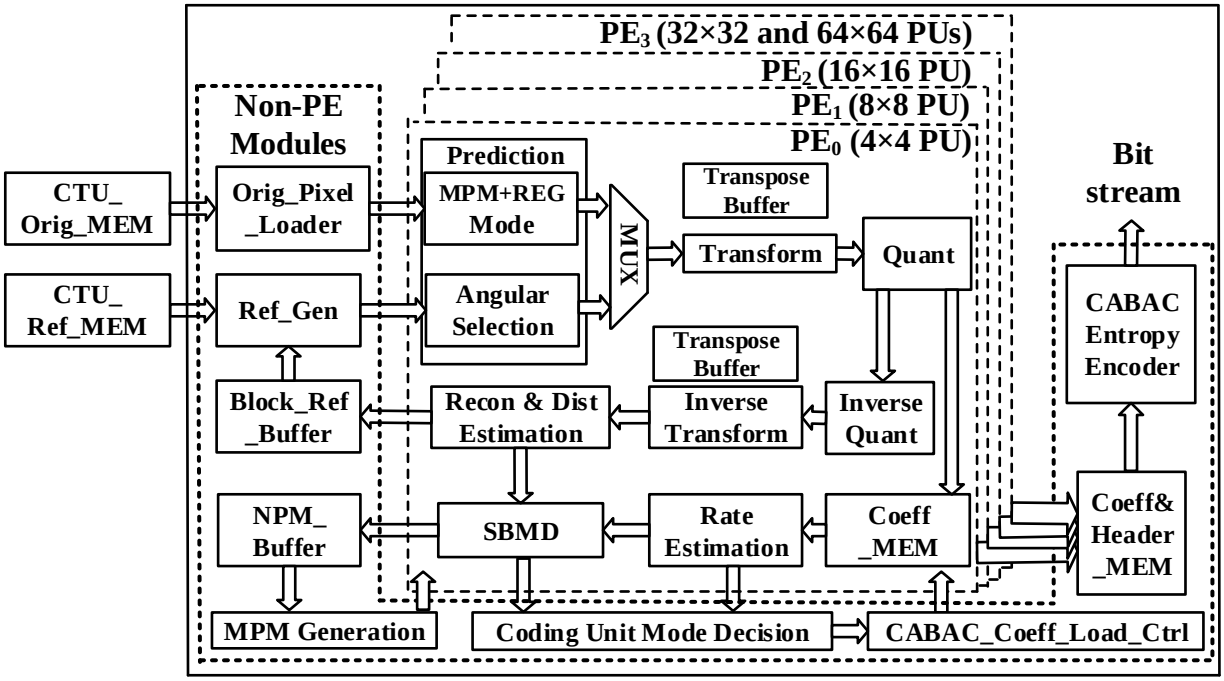
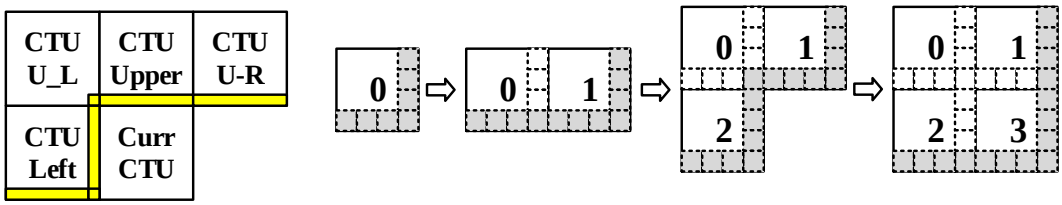


Fig. 4.1. Hardware architecture of the proposed fully-parallel H.265/HEVC intra encoder.

4.3.1 Design Details of Non-PE Modules

1) Original Pixel Loading and Reference Pixel Generation: Two on-chip memory blocks (CTU_Orig_MEM and CTU_Ref_MEM) are fed to the proposed intra encoder. As the proposed design supports 4:2:0 video format, a CTU contains one 64×64 luma block and two 32×32 chroma blocks. Loading CTUs from an external memory into the CTU_Orig_MEM is beyond the scope of this chapter. The CTU_Orig_MEM stores original pixels of two CTUs, which are 12288 8-bit pixels. Thus, the size of CTU_Orig_MEM is 12 KB. The reference pixels of CTUs are stored in CTU_Ref_MEM. As shown in Fig. 4.2(a), all pixels in the yellow region should be

stored in the CTU_Ref_MEM before the current CTU begins to process. The number of CTU reference pixels is determined by video resolution. For example, 3840+64 reference pixels are stored for luma prediction in 4K videos. Assume the video format is 4:2:0, the number of reference pixels for each chroma component in 4K videos is 1920+32. As a result, the required size of CTU_Ref_MEM is about 7.6 KB.



(a) CTU reference pixels (b) Block reference pixels inside 4 4×4 PUs of an 8×8 CU

Fig. 4.2. *CTU_Ref_MEM* for (a) reference pixels of a current CTU, (b) reference pixels in four 4×4 PUs of an 8×8 CU.

In order to minimize latency caused by loading pixels from *CTU_Orig_MEM*, the *Orig_Pixel_Loader* module is designed to load 16 pixels per clock cycle and store them in an internal 32×32 register array. These loaded pixels are fed to 4 PEs for intra prediction. The use of *Orig_Pixel_Loader* increases the throughput of intra prediction. The *Block_Ref_Buffer* module stores reference pixels of prediction blocks inside a CTU. Fig. 4.2(b) illustrates the process sequence of four 4×4 PUs in an 8×8 CU and corresponding pixels to be stored as reference pixels. For example, for the first 4×4 PU with index 0, 7 pixels need to be stored for intra prediction of the successive PUs. As shown in Fig. 4.2(b), the maximum number of reference pixels that need to be stored in the *Block_Ref_Buffer* module for 4×4 PUs is 15. Similarly, the number of pixels to be stored for other luma PU sizes can be deduced and listed in Table 4.2. Particularly, since a 64×64 luma PU is performed based on four 32×32 luma PUs, the temporal prediction results of each 32×32 luma PU are stored. The final mode decision of a current CTU leads to updating the CTU reference pixels in *CTU_Ref_MEM*. The right-column pixels of a

current 64×64 CTU are used as reference pixels for the next CTU on its right side, while the bottom-row pixels are referred by CTUs below the current 64×64 CTU line. Hence, a total number of 127 pixels are updated into *CTU_Ref_MEM*. Regarding the register requirements for chroma PUs, each chroma PU needs the same number of reference registers as luma PUs except 32×32 PUs. In a 64×64 CTU, there are four 32×32 luma PUs, while it has only one 32×32 chroma U (Cb) block and one 32×32 chroma V (Cr) block. Thus, for each 32×32 chroma block, it only needs 63 registers to store reference pixels. As 4:2:0 video format is used in this work, there is no 64×64 chroma PUs involved.

Table 4.2 Required number of registers in Block_Ref_Buffer.

PU Size	Luma (Y)	Chroma (Cb)	Chroma (Cr)
4×4	15	15	15
8×8	31	31	31
16×16	63	63	63
32×32	127	63	63
64×64	127	N/A	N/A

2) MPM Generation: The PU mode decision selects the best prediction mode among all candidates. The selected mode of a current PU is used to generate three Most Probable Modes (MPMs) for subsequent PUs. Since the smallest PU size is 4×4, 256 prediction modes are stored in a 64×64 CTU for MPM generation. Its circuit implementation may be an on-chip memory or register array with 256 contents. In this work, an efficient storage scheme is proposed to largely reduce memory/register array size for prediction modes.

Fig. 4.3 illustrates two examples of referred prediction modes for MPM generation. Instead of 4 and 16 prediction modes, 3 and 7 prediction modes are involved for MPM generation of successive PUs in 8×8 and 16×16 regions, respectively. Similarly, storing 15 and 31 prediction modes are sufficient for MPM generation of 32×32 and 64×64 regions, respectively. In addition, since prediction modes from the top CTU are not used in MPM

generation, 16 prediction modes from the left CTU need be stored. Based on the above observations, a proposed neighboring PU mode (NPM) buffer can efficiently store these prediction modes. By considering all prediction modes from different PU sizes, it requires 72 registers for MPM generation. This number is 77% less than the normal approach, where all prediction modes of 4×4 blocks in a CTU are stored and hence it needs 256+16 registers for MPM generation. Once the upper and left prediction modes are derived, three MPMs for a current PU can be determined according to the H.265/HEVC standard. The proposed MPM generator provides three MPMs for each PU and sends them to corresponding PEs in Fig. 4.1.

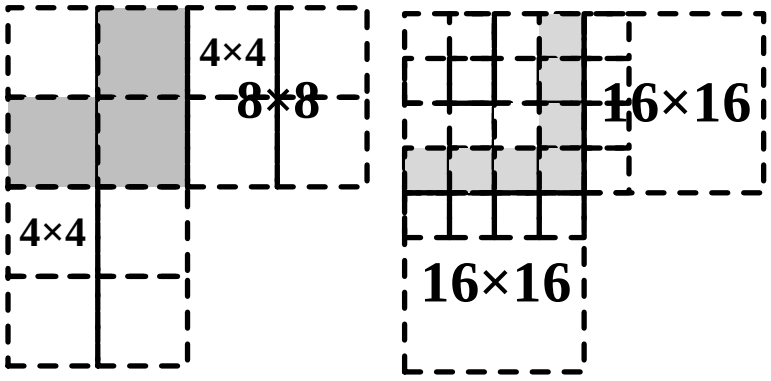


Fig. 4.3. Referred prediction modes for MPM generation with respect to (left) 8×8 region and (right) 16×16 region.

3) Coding Unit Mode Decision: As an indispensable process in H.265/HEVC intra prediction, RDO tries to make the finest balance between compression ratio and video quality. Based on estimated bit rate and distortion, the Lagrangian function is employed as the criteria to find the best prediction mode for each PU. CUs are repeatedly divided into four sub-CUs to search the best partition. The high computational complexity and long processing time prevent full RDO implementation. Therefore, we propose a simplified and hardware-friendly RDO scheme. The RDO scheme consists of two parts: one part is for PU mode selection and the other is for CU size optimization. The RDO scheme for PU mode selection will be elaborated in

section 4.3.2. Here, we focus on the description of CU mode decision (CUMD) in the partition optimization process.

Fig. 4.4 depicts the CUMD process that relies on the RD cost calculated from four PEs. For an 8×8 CU, the corresponding PU partition could be an 8×8 PU or four 4×4 PUs. Hence, the RD costs of four 4×4 PUs and an 8×8 PU are compared. Then, the output flag "*part_size*" indicates the 8×8 CU mode decision result. Meanwhile, this 8×8 RD cost, which is the smaller one from the RD cost of four 4×4 PUs and the RD cost of an 8×8 PU, will be used to compare with the RD cost of a 16×16 PU. Likewise, the RD cost of remaining three 8×8 CUs in this 16×16 pixel region can be deduced. The total RD cost of four 8×8 CUs is compared with the RD cost of 16×16 CU. Then, an output flag "*split_16×16*" is generated for a 16×16 CU. Similarly, two output signals "*split_32×32*" and "*split_64×64*" are generated in the CUMD with respect to 32×32 and 64×64 CUs. These output flags indicate the partition of this CTU.

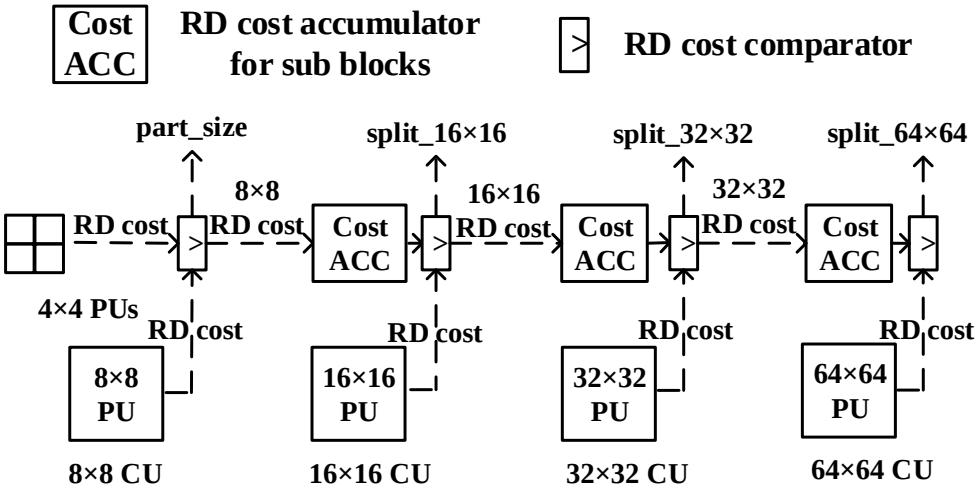


Fig. 4.4. CU mode decision in partition optimization through RD cost comparison

CU mode decision also affects reference pixel selection for the current block and prediction modes update in the NPM buffer for MPM generation. For example, if the output flag "*part_size*" is 1, 4×4 PUs are chosen for reference pixel selection. Thus, the reference pixels for the successive PUs are derived from 4×4 PUs instead of an 8×8 PU. In addition, prediction

modes of 4×4 PUs are stored for MPM generation of successive PUs. Reference pixels and prediction modes for 16×16, 32×32, and 64×64 CUs are determined based on the output flags "*split_16×16*", "*split_32×32*", and "*split_64×64*", respectively.

4) CABAC Bit Stream Generation: Due to its superior coding efficiency, context-adaptive binary arithmetic coding (CABAC) is the only adopted entropy coding algorithm in H.265/HEVC standard. It encodes binary symbols that are derived from quantized coefficients and prediction information. As shown in Fig. 4.1, under the control of the CABAC_Coeff_Load_Ctrl module, quantized coefficients from the Coeff_MEM module along with prediction header information are loaded into the Coeff&Header_MEM module. In order to realize pipelined operations between intra prediction and CABAC encoder, the Coeff&Header_MEM module is designed with the capacity of storing data of two 64×64 CTUs. Therefore, the memory size of Coeff&Header_MEM is 31 KB.

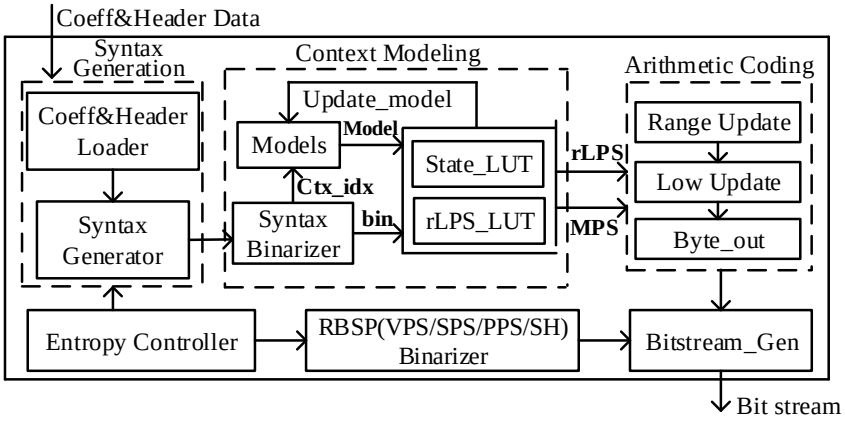


Fig. 4.5. Hardware architecture of CABAC entropy encoder

Fig. 4.5 plots the hardware architecture of proposed CABAC entropy encoder, which consists of four main functional blocks (*i.e.*, Syntax Generation, Context Modeling, Arithmetic Coding, and Raw Byte Sequence Payload (RBSP) binarizer). Activated by the entropy controller, the Syntax Generation block reads quantized coefficients and header information of a specific

CU. Then, based on quantized coefficients and header information, syntax elements are generated and sent to the *Syntax Binarizer*.

Syntax Binarizer conducts binarization for each syntax element and produces binary symbols (*i.e.*, bin). In a CABAC entropy encoder, there are two types of bins involved: regular coded bin and bypass coded bin. The former is coded based on context modeling and binary arithmetic coding, while the latter does not rely on context modeling. For a regular coded bin, the corresponding context model is read using a signal "Ctx_idx". The new context model "Update_model" for this regular coded bin is generated from the state transition table (*i.e.*, State_LUT) and bin value. "Update_model" will replace the context model contents in the *Models* module. The Arithmetic Coding block involves three steps, including range update, low update, and byte packing. In the Context Modeling block, a variable "rLPS" is determined by bin value, context model and range value. The new range value and low value are computed using current bin value, current "rLPS" value and the most probable symbol. According to the low value and its updating status, a byte is generated in the "Byte_out" process. The *RBSP Binarizer* module contains basic coding parameter sets, including video parameter set (VPS), sequence parameter set (SPS), picture parameter set (PPS), and slice header (SH). These parameter sets are processed in this module to generate the byte sequence that conforms to HEVC standard. The *Bitstream_Gen* block combines the byte sequence from the Arithmetic Coding block and RBSP from the *RBSP Binarizer* to output a final bit stream. The proposed CABAC encoder runs at a maximum frequency of 720 MHz and shows a throughput of 850 Mbins per second.

4.3.2 Design Details of PE Modules

PE modules are described and discussed in this section. PE_0-PE_3 are dedicated to 4×4, 8×8, 16×16, and 32×32 PUs/TUs, respectively. As a 64×64 PU prediction is based on 32×32

PUs/TUs, 64×64 PUs are embedded into PE_3 to save hardware resources. When reference pixels are ready in the *CTU_Ref_MEM* module, the process of intra prediction starts. Each PE performs a series of computations such as intra prediction, transformation, quantization, inverse quantization, inverse transformation, reconstruction, CABAC rate estimation, distortion estimation, and simplified block mode decision. Original pixels and prediction pixels are stored in on-chip memories for reconstruction and distortion calculation. Residuals of prediction block are used for transformations. A transpose buffer stores coefficients after each horizontal transformation. Coefficients after each vertical transformation are used for quantization. Quantization coefficients are stored for rate estimation and entropy coding. Residuals from inverse transformation are used for frame reconstruction and distortion calculation.

1) Intra Prediction: Efficient hardware architectures are required to implement the proposed algorithm adaptations in PU mode preselection. As the PU chroma mode preselection only involves four regular modes, its hardware architecture is straightforward. Therefore, we focus on hardware architecture explorations for PU luma mode preselection, which involves complex choices in both intra prediction and RDO process.

Table 4.3 Luma prediction and RDO modes in the proposed design.

PU size	Prediction Modes	RDO Candidate Modes
64×64	1 MPM	1 MPM
32×32	35	3 MPM + 1 Hadamard
16×16	35	3 MPM + 1 Regular + 1 Hadamard
8×8	35	3 MPM + 1 Regular + 1 Hadamard
4×4	3 MPMs + 4 Regular + 12 Angular	3 MPM + 4 {Regular + Hadamard}

As shown in Table 4.3, PU luma prediction modes and RDO candidate modes consist of MPMs, regular modes, and Hadamard modes. In order to balance the overall throughput of all PEs, these mode choices are different from each PU size. For example, only one MPM is

selected for 64×64 PUs, while three MPMs are selected for other PU sizes. For 8×8 and 16×16 PUs, one regular mode from DC, Planar, Vertical, and Horizontal modes is adopted as RDO candidate modes. For 4×4 PUs, the number of angular modes for RDO process is determined by the number of common mode between MPMs and regular modes. In case of only one common mode, the angular mode with the smallest Hadamard cost is adopted as RDO candidate mode, and the other two angular modes are discarded. For 8×8~32×32 PUs, in addition to MPMs and regular modes, one Hadamard mode with the smallest Hadamard cost is selected from 35 prediction modes as an RDO candidate mode.

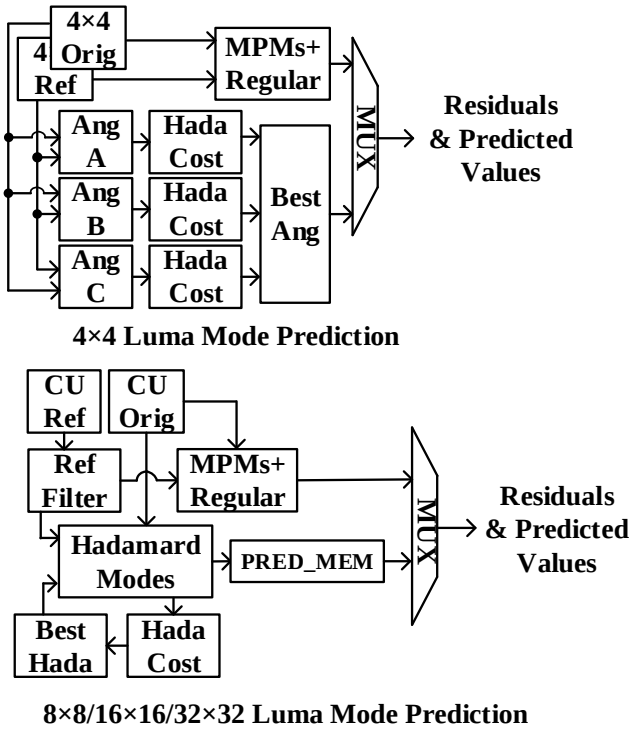


Fig. 4.6. Intra prediction architecture: (a) for 4×4 PUs, (b) for other size PUs.

Fig. 4.6 depicts the proposed intra prediction architectures for all PU blocks. For 4×4 blocks, selected MPMs and regular modes are processed immediately when intra prediction starts. Yet, three angular modes are determined by the Hadamard cost. In Fig. 4.6(a), to accelerate the derivation of the best angular modes for 4×4 PUs, three dedicated angular

prediction modules are implemented. Then, through Hadamard cost comparison, one angular mode with minimum Hadamard cost is selected as the best angular mode. In order to improve throughput, prediction of each angular mode takes one clock cycle for 4×4 PUs. Fig. 4.6(b) shows the intra prediction architecture of other PU blocks. A reference filter is included for reference smoothing, and its operation varies with prediction modes and block sizes. Once the best Hadamard mode is determined, the corresponding prediction results are stored in the PRED_MEM memory for the subsequent RDO process. After MPMs and regular modes are evaluated, the prediction result of the best angular mode is provided for transformation. From 8×8 ~ 32×32 blocks, each prediction block is predicted on a row-by-row basis. Therefore, 8, 16, and 32 clock cycles are required to predict each mode in 8×8, 16×16, and 32×32 blocks, respectively.

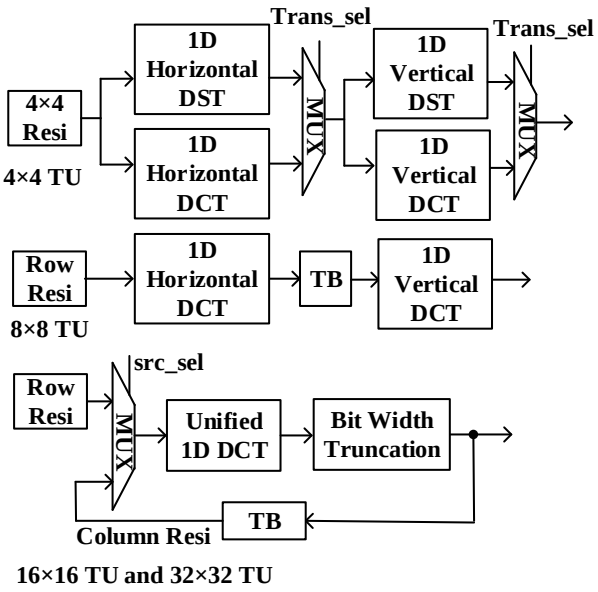


Fig. 4.7. Transform Architecture for 4×4, 8×8, 16×16, and 32×32 TUs.

2) Transform Coding and Quantization: DST is only used for luma 4×4 TUs, while DCT is adopted for the remaining TUs. As DCT and IDCT are closely associated with computational complexity in an H.265/HEVC hardware implementation [27], it is necessary to investigate high-

throughput and area-efficient hardware architectures for DCT/IDCT implementation.

Fig. 4.7 shows the proposed transform architectures for all TUs. In order to improve throughput, two-stage pipelines are used for transforms of 4×4 and 8×8 TUs [47]. Note 4×4 TUs involve both DCT and DST transforms. Based on the color component of current 4×4 TU, a signal "*Trans_sel*" selects the output of 1D horizontal transform and passes it to 1D vertical transform. For 4×4 TUs, 1D horizontal/vertical transformation takes one clock cycle. DCT Transforms of 8×8 TUs are involved in a two-stage hardware, which includes a transpose buffer (TB) to store intermediate coefficients. For 16×16 and 32×32 TUs, in order to reduce computational resources, a unified 1D DCT structure is proposed. A control signal "*src_sel*" selects input data source. A horizontal transformation of residuals is first performed on a row-by-row basis. The intermediate results are stored in a TB. After the completion of a horizontal transformation, intermediate results are inputted to the unified 1D DCT again for a vertical transformation. Due to different input data sizes in horizontal and vertical transformations, a bit width truncation block is added to ensure correctness of transformation. The throughput and required number of clock cycles for each TU are summarized in Table 4.4.

Table 4.4 Throughput and required clock cycles of transformation.

TU size	Throughput (pixels/cycle)	Required Clock Cycles
32×32	16	64
16×16	8	32
8×8	8	8
4×4	16	1

3) Highly-Parallel Table-Based CABAC Rate Estimator: This section describes the hardware design of table-based CABAC rate estimator. CABAC based rate estimation algorithm leads to the best accuracy, so it is adopted in the HEVC HM-15.0 software. However, due to the sequential data/timing dependency of syntax element coding, the throughput of hardware

implementations of CABAC based rate estimation is very low. To address this low-throughput challenge and retain the most accurate estimation, a highly-parallel architecture is proposed to alleviate data/timing dependency in rate computation, where the involved syntax elements are classified into five independent syntax processing groups in [26]. In order to reduce computational complexity, the proposed CABAC rate estimator is only applied to luma blocks, while non-zero quantized coefficients are used for rate estimation of chroma blocks. The proposed luma rate estimator uses look-up tables to perform context modeling and updating. Multiple rate estimate instances are implemented for high parallelism along with a global context buffer. To keep the same rate estimation throughput for four PEs, various numbers of rate estimate instances are used as shown in Table 4.5.

Table 4.5 Number of rate estimator instances for each PE.

PE	Number of RDO Candidates	Number of Rate Estimate Instances
0	7	7
1	5	2
2	5	2
3	4 for 32×32, 1 for 64×64	1 for 32×32, 1 for 64×64

To accelerate rate estimation, five independent syntax groups in a rate estimate instance operate simultaneously. In Fig. 4.8, the coefficient processor takes input coefficients to generate five syntax groups. Syntax groups A-D contain regular coded bins and bypass coded bins, while group E contains only bypass coded bins. During rate estimation, as regular coded bins depend on context models, local context buffers for syntax groups A-D are implemented inside each rate estimate instance, and each local context buffer serves to a dedicated syntax group. At the beginning of rate estimation, the latest global context models of a current CU are loaded into corresponding local context buffers in each rate estimate instance. This operation is called the context model localization. Then, based on bin value and its context model of each regular coded

bin, a new context model is derived from the corresponding lookup table. Local context models inside each rate estimate instance will be updated accordingly, while the global context models still remain unchanged. In order to minimize latency related to context model loading and updating, these local context buffers are implemented by registers instead of on-chip memories. As the syntax group E only contains bypass coded bins, its rate estimation is approximated by bin counter and Exp-Golomb coding that performs non-zero coefficient binarization. Finally, all fractional rates of five syntax groups are accumulated as the total bit rate. Since each rate estimation instance has its own local context model memory, the conflicts of context modeling between different instances can be avoided. Once the rate estimation is complete for all instances, RD mode and size decision is made. Based on the decision result, the updated local context models that correspond to the best CU mode and partition among both rate estimation instances are selected to update the global context models. In this way, our proposed procedures of context model loading and updating prevent potential context model conflicts among multiple rate estimator instances. The global context model buffer is synchronously updated with CABAC rate estimators.

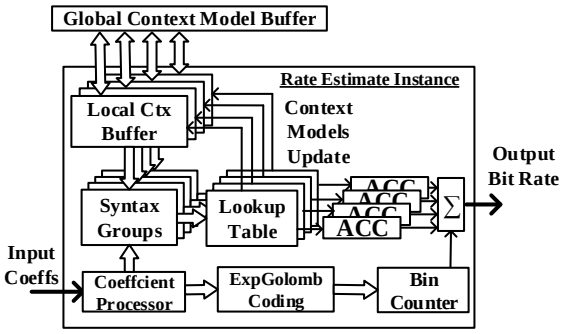


Fig. 4.8. Proposed highly-parallel table-based rate estimator.

4) Simplified Block Mode Decision (SBMD): This module determines the best prediction mode for luma and chroma blocks. Mode decision process is independent for luma and chroma

blocks. For luma prediction blocks, RD cost of each prediction mode is calculated when its corresponding distortion and rate are available. Then, a prediction mode with minimum RD cost is selected as the best prediction mode. On the other hand, for chroma prediction blocks, the rate is approximated using non-zero quantized coefficients. RD cost of a chroma mode is derived when its distortion is available. This simplification greatly reduces hardware complexity and increases throughput. Since each PU contains two chroma blocks, the distortion and rate of two chroma blocks are added to the total RD cost. Once the best prediction mode is determined by the SBMD module, its total RD cost is sent to the CUMD block for CU mode decision.

4.3.3 Timing Diagram of Proposed Intra Prediction

Fig. 4.9 shows the parallel processing of various PUs in four PEs and their data/timing dependency. Once original CTU pixels and reference pixels are ready, all PEs start processing. Since each 8×8 CU contains 4×4 and 8×8 partitions, two PEs handle 8×8 CUs. Specifically, PE_0 and PE_1 are responsible for 4×4 and 8×8 PUs, respectively. After PE_0 and PE_1 determine the best prediction modes, the *CUMD* module is activated to find the best partition for this 8×8 CU. This procedure is repeated four times to obtain the best RD cost of sub-CUs of a 16×16 CU. Meanwhile, 16×16 PUs are processed in PE_2 and its best RD cost is calculated. After completing the mode decision of this 16×16 CU and its sub-CUs, the *CUMD* module is activated to determine the optimal partition of this 16×16 block. 32×32 and 64×64 CUs are processed in a similar manner.

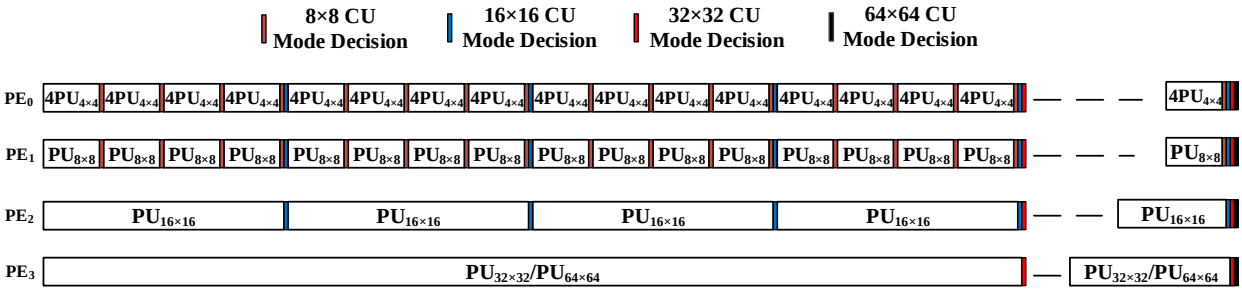


Fig. 4.9. Parallel processing of various PUs in four PEs and their data/timing dependency.

Fig. 4.10 plots the timing diagrams of 4×4 PUs in PE_0 and 8×8 PUs in PE_1. This figure includes both luma and chroma blocks. For an 8×8 PU, five luma prediction (*i.e.*, three MPMs, one regular mode, and one angular mode) are processed one by one in PE_1. It takes 8 clock cycles to process a luma prediction mode. Because intra prediction is performed on a row-by-row basis, horizontal transform coding (T_1D) is performed one clock cycle after intra prediction. It takes 8 clock cycles for vertical transform coding (T_2D), which starts one clock cycle after the completion of T_1D. Quantization (Quant) begins after T_2D and it is followed by inverse quantization (iQuant). Once all the transform coefficients are quantized, CABAC based rate estimation begins immediately. To increase rate estimation throughput of 8×8 PU intra prediction, 2 rate estimate instances are implemented for PE_1. For each luma mode, it takes 32 clock cycles in PE_1 to complete computation from mode prediction to distortion calculation. Due to a fine pipelined design, the total number of clock cycles for processing five luma prediction modes is 68. It represents an average of 14 clock cycles per mode. The throughput of each pipeline stage is 8 pixels/cycle in PE_1. After completing 5 luma mode prediction for an 8×8 CU, intra prediction for one chroma mode of 16×16 CU starts in PE_1. Note the chroma block size is 8×8 in 16×16 CUs. Chroma mode processing of 16×16 CUs is treated as 8×8 sub-CUs in luma mode processing. Chroma mode processing of 8×8 CUs is scheduled after 4×4 PUs in PE_0.

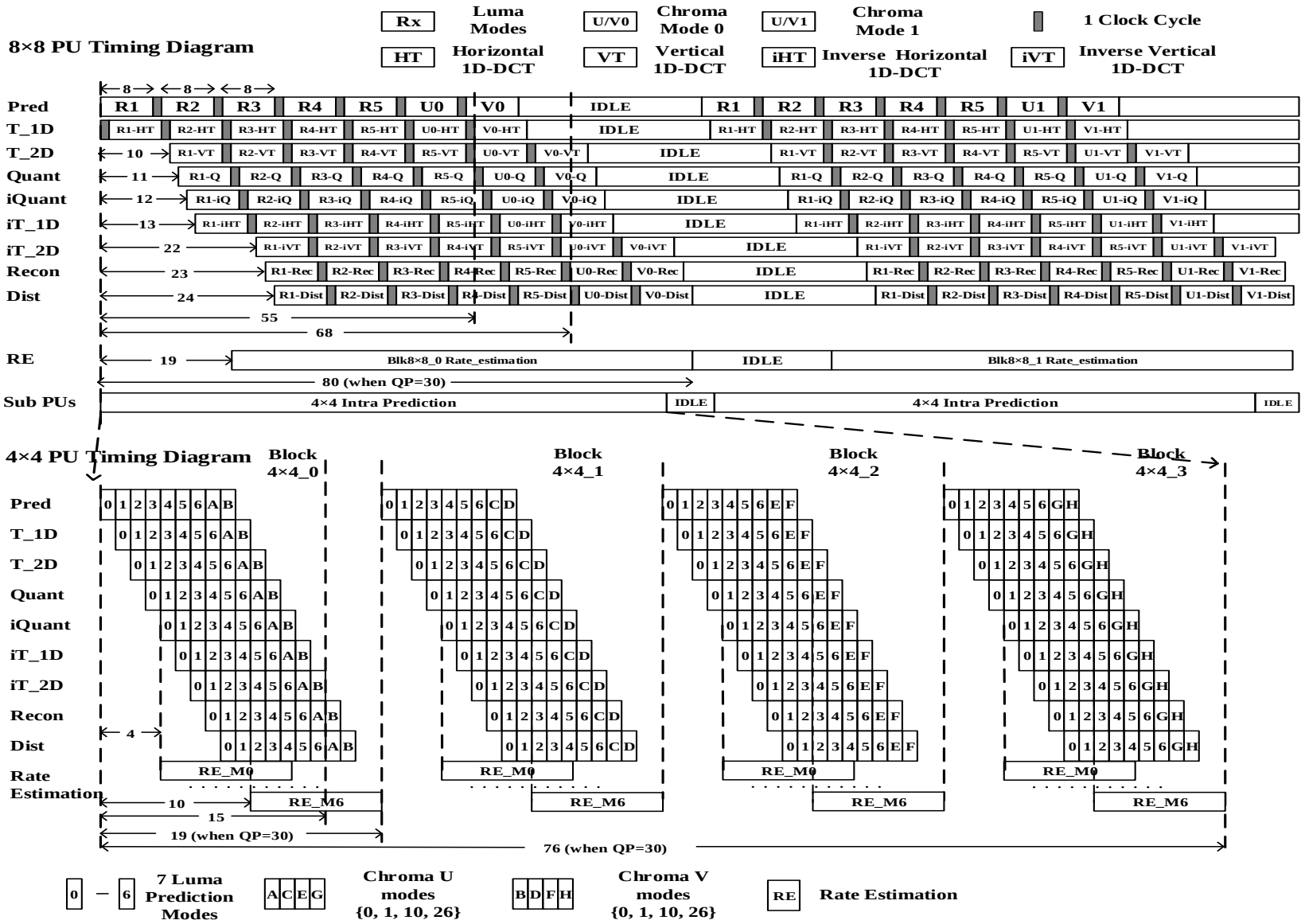


Fig. 4.10. Timing diagrams of 4×4 PUs in PE_0 and 8×8 PUs in PE_1.

Intra prediction of 4×4 PUs in PE_0 is similar to that of 8×8 PUs in PE_1. Four 4×4 PUs are contained in an 8×8 CU, each PU goes through an RDO process to evaluate 7 preselected luma prediction modes. Three MPMs are processed first, followed by the other four selected modes described in the previous section. As plotted in Fig. 10, it takes 1 clock cycle to process a luma mode prediction for 4×4 PUs. Thus, a pipelined hardware can achieve a throughput of 16 pixels per clock cycle. In Fig. 4.10, symbols (0-6) stand for seven luma modes of 4×4 PUs, while symbols (A, C, E, and G) and symbols (B, D, F, and H) represent mode 0 (Planar mode), mode 1 (DC), mode 10 (Horizontal), and mode 26 (Vertical) for chroma U and V, respectively. For 4×4 PUs, it requires 15 clock cycles from prediction to distortion calculation of 7 luma modes. In order to improve the rate estimation throughput of 4×4 PUs, 7 rate estimate instances are implemented in PE_0, so each instance deals with one prediction mode. Rate estimation starts immediately when quantized coefficients of each prediction mode are ready. RD cost of each mode is calculated based on distortion and rate. The smallest RD cost is selected in the luma mode decision.

Next, we discuss the overall throughput of 4×4 and 8×8 PUs. As the time required for CABAC based rate estimator depends on QP, we assume QP is 30 in the following analysis. When QP is set to 30, our experimental studies show it takes an average of 9 and 25 clock cycles for rate estimation of a 4×4 and 8×8 PU, respectively. For 4×4 PUs, as quantization of the 7^{th} prediction mode is done after the 10^{th} clock cycle, 19 clock cycles are enough to complete the computation from prediction to rate estimation. Thus, the total number of clock cycles for four 4×4 PUs is 76, which indicates that the RD cost of sub-PUs of an 8×8 CU is derived within 76 clock cycles. For 8×8 PUs, since quantization of the 5^{th} luma mode is done after the 55^{th} clock cycle, 80 clock cycles are enough to complete the computation from prediction to rate

estimation. This number is close to 76 of four 4×4 PUs. When RD costs of both 8×8 CU partitions are available, an RD cost comparison is performed. Thus, the best CU/PU combination and prediction mode are determined.

The nature of CABAC algorithm is context-based entropy coding, so its processing time for rate estimation is not fixed. The maximal and minimal numbers of clock cycles in CABAC rate estimation heavily depend on the design choices (prediction mode set, quantization parameter, parallelism level, hardware implementation architecture, etc.) and the specific video sequences under test. In practice, due to the content variation of video sequences, some blocks require more clock cycles to process, while other blocks require fewer clock cycles. These blocks requiring fewer cycles compensate delays introduced by complex blocks. If there are many blocks requiring more clock cycles, the average throughput is degraded, so the intra encoder can only support a lower frame rate or a less resolution of video encoding applications.

4.4 EXPERIMENTAL IMPLEMENTATION AND RESULTS

The proposed algorithm adaptations have been implemented in a modified HM-15.0 software. Table 4.6 provides the algorithm-level coding performance comparison between existing designs and this work. All the results in Table 4.6 are obtained with the disabled RQT. The reference [28] supports a maximum resolution of 1080P, if we compare the coding performance in only 1080P sequences (*i.e.*, class B), the reference [28] provides an average increase of 4.62% in BD-Rate, while our work provides an average increase of 3.01% in BD-Rate. The design [29] and this work provide complete results for classes A-E and 4K video sequences except for Beauty. The BD-Rate increase in the reference [29] is 5.94%, while it is 4.36% in this work.

The proposed fully-parallel hardware architecture of H.265/HEVC intra encoder has been

implemented in Verilog and synthesized by field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) technology. The FPGA implementation is based on Altera Stratix V GX, and the ASIC implementation uses TSMC 90nm technology. The maximum operating clock frequencies for FPGA and ASIC implementations are 120MHz and 320MHz, respectively.

Table 4.6 Comparison of experimental results of intra encoding algorithms.

		Zhu [28]		Pastuszak [29]		This work	
Class	**Sequences**	**BD_Rate [%]**	**BD_PSNR [dB]**	**BD_Rate [%]**	**BD_PSNR [dB]**	**BD_Rate [%]**	**BD_PSNR [dB]**
A	People On Street	4.61	-0.21	5.18	-0.25	3.31	-0.17
A	Traffic	4.34	-0.21	4.73	-0.22	4.49	-0.22
B	Park Scene	3.39	-0.11	3.97	-0.15	2.04	-0.09
B	Kimono	4.39	-0.12	3.42	-0.11	2.92	-0.09
B	Basketball Drive	6.73	-0.17	7.39	-0.20	4.33	-0.13
B	BQ Terrace	4.32	-0.19	4.99	-0.20	1.95	-0.09
B	Cactus	4.28	-0.14	5.70	-0.18	3.87	-0.14
Average B (1080P HD)		4.62	-0.15	5.09	-0.17	3.02	-0.11
C	Basketball Drill	4.63	-0.21	8.91	-0.39	12.12	-0.59
C	BQ Mall	4.15	-0.20	6.23	-0.30	3.30	-0.17
C	Party Scene	N/A	N/A	5.39	-0.34	2.76	-0.21
C	Race Horses	3.38	-0.19	5.74	-0.29	4.97	-0.29
D	Basketball Pass	4.80	-0.24	7.28	-0.39	6.02	-0.37
D	Blowing Bubbles	3.44	-0.19	5.91	-0.35	3.63	-0.27
D	BQ Square	1.97	-0.15	6.41	-0.42	2.37	-0.18
D	Race Horses	N/A	N/A	6.78	-0.36	6.12	-0.40
E	Kristen And Sara	5.86	-0.24	7.88	-0.36	6.28	-0.26
E	Four People	N/A	N/A	5.79	-0.29	3.33	-0.16
E	Johnny	5.15	-0.21	8.03	-0.30	5.31	-0.19
F	Slide Editing	N/A	N/A	N/A	N/A	2.48	-0.36
F	Slide Show	N/A	N/A	N/A	N/A	6.72	-0.59
F	China Speed	N/A	N/A	N/A	N/A	4.12	-0.37
4K	Beauty	N/A	N/A	N/A	N/A	5.06	-0.08
4K	Bosphorus	N/A	N/A	6.03	-0.16	6.19	-0.16
4K	Honey Bee	N/A	N/A	5.35	-0.08	2.31	-0.04
4K	Jockey	N/A	N/A	7.31	-0.10	4.54	-0.08
4K	Ready Steady Go	N/A	N/A	5.37	-0.16	4.84	-0.11
4K	Shake & Dry	N/A	N/A	2.58	-0.06	1.84	-0.04
4K	YachtRide	N/A	N/A	6.11	-0.13	5.79	-0.16
Average 4K except for Beauty		N/A		5.46	-0.12	4.25	-0.10
Average 1080P HD and 4K except for Beauty		N/A		5.29	-0.14	3.69	-0.10
Average value for A-E except for Party Scene, Race Horses, and Four People		4.36	-0.19	N/A		4.46	-0.22
Average for A-E and 4K except for Beauty		N/A		5.94	-0.24	4.36	-0.19

Table 4.7 shows the memory usage for storing original pixels, prediction pixels,

quantization coefficients and reconstruction pixels in each PE, as well as memory demand for non-PE modules. Denoted as "bank×column×width", the memory size includes both luma and chroma components. *ORIG_MEM* and *PRED_MEM* store original pixels and prediction pixels after intra prediction, respectively. *QUANT_MEM* stores quantization coefficients, which are the output of intra prediction. *REC_MEM* stores reconstructed pixels that are reference pixels for future blocks. In order to reduce the data access delay in PE_0, all computational results are stored in registers to improve the throughput, so no memory blocks are included in PE_0. The total memory in all four PEs is 563 Kbits (*i.e.*, 69 KB). As introduced in section 4.3.1, memory blocks (*CTU_Orig_MEM*, *CTU_Ref_MEM*, *Coeff&Header_MEM*) serve all PEs in this hardware architecture. The sizes of these memories are 12 KB, 7.6 KB, and 31 KB, respectively. Therefore, the required total amount of memory resources is 120 KB for the entire H.265/HEVC intra encoder as well as the external original and reference pixel memories.

Table 4.7 Memory usage in the proposed intra encoder (unit: bit).

	ORIG_MEM	PRED_MEM	QUANT_MEM	REC_MEM	Total(PE)
PE_0	Register	Register	Register	Register	None
PE_1	2×24×32	2×40×32	2×104×64	1×104×64	24,064
PE_2	4×48×32	4×80×32	4×720×64	2×720×64	292,864
PE_3	8×32×32	8×160×32	8×256×64	4×256×64	245,760
Total (PE)	15,872	53,760	328,704	164,352	562,688
Non-PE	Coeff&Header_MEM				256,960
Total	819,648				

The processing time of our proposed CABAC based rate estimator depends on QP values and PU sizes. Table 4.8 lists the required average, maximum, and minimum numbers of clock cycles for rate estimation with different PU sizes and QP values. A smaller QP leads to better video quality after compression, but the resultant rate estimation throughput is lower due to more

quantization coefficients are involved in binarization and context-adaptive probability modeling. For each PU size with four QP options (22, 27, 32, and 37), the maximum number of clock cycles is obtained from the worst cases, which correspond to the largest number of non-zero quantization coefficients for CABAC rate estimation, and the minimum number of clock cycles represents the best cases, which correspond to zero or very few non-zero quantization coefficients for CABAC rate estimation. Since the PUs requiring fewer clock cycles compensate delays introduced by PUs requiring more clock cycles, so the average number of clock cycles determines the overall rate estimation throughput. The average numbers in Table 4.8 are sufficient to meet the system throughput requirements (*i.e.*, 1080P @ 45fps in FPGA, and 4K @ 30fps in 90nm ASIC) in this work.

Table 4.8 Average, maximum, and minimum clock cycles of rate estimator vs. QP value and PU size.

QP\PU	4×4	8×8	16×16	32×32	64×64
22	11.7	44.7	160.6	608.3	2421.3
27	9.8	27.8	99.2	391.0	1563.6
32	8.7	22.5	61.6	235.4	944.5
37	8.0	18.1	40.3	145.6	587.1
Max for above QPs	22	99	340	1292	5112
Min for above QPs	7	8	6	6	25

Table 4.9 Average number of clock cycles and throughput of intra prediction vs. QP value and PU size.

QP\PU	4×4		8×8		16×16		32×32	
	CLK	THP	CLK	THP	CLK	THP	CLK	THP
22	21.7	0.7	99.7	0.6	326.6	0.8	938.3	1.1
27	19.8	0.8	82.8	0.8	265.2	1.0	720.9	1.4
32	18.7	0.9	77.5	0.8	227.6	1.1	565.4	1.8
37	18.0	0.9	73.1	0.9	206.3	1.2	475.6	2.2

Table 4.9 shows the required average number of clock cycles and throughput (*i.e.*, THP) for intra prediction with different PU sizes and QP values. Since a 64×64 PU is split into four 32×32 PUs for prediction, a 64×64 PU is regarded as four 32×32 PUs in the throughput

evaluation. In Table IX, intra prediction of 8×8 PUs has the lowest throughput for all QP values. For example, the lowest throughput is 0.6 for a QP value of 22. Hence, intra prediction of 8×8 PUs is the throughput bottleneck of proposed highly-parallel hardware architecture. For a 64×64 CTU, the required number of clock cycles is calculated by 4096/THP. Assume the QP is 27, the minimum throughput 0.8 is obtained from the Table 4.9. Thus, 5120 clock cycles are required for intra prediction of a 64×64 CTU. Considering the reported clock frequency at our FPGA design (*i.e.*, 120MHz), this proposed architecture supports up to 1080P @ 45 fps. Since our ASIC implementation in TSMC 90nm technology supports 320MHz clock frequency, the throughput of this architecture can sustain up to 4K@ 30 fps.

Table 4.10 Resource comparison of proposed design implemented by FPGA and TSMC 90nm technology.

Module	Stratix V GX (# of ALUT + # of DSP)				TSMC 90nm (K gate)			
	PE_0	PE_1	PE_2	PE_3	PE_0	PE_1	PE_2	PE_3
Intra Prediction	7019 + 64	12840 + 64	19795 + 82	25717 + 106	68.2	103.0	129.4	183.0
Trans	2568 + 56	1443+ 24	3133 + 50	10611 + 224	51.4	32.5	72.5	269.9
Quant	1129 + 16	583 + 8	1321 + 16	2497+ 32	28.4	14.4	29.1	61.1
iQuant	368 + 16	184+ 8	368 + 16	704 + 32	27.4	13.6	27.1	54.2
iTrans	3820 + 56	1781 + 24	3526 + 52	12068 + 222	52.4	30.1	67.3	258.2
Reconstruction	439	217	673	744	10.4	0.8	1.6	3.2
Distortion Calculation	401 + 12	197 + 6	410 + 12	817 + 24	8.8	4.6	9.2	18.2
Rate Est & SBMD	12620	8107 + 5	5667 + 6	7970 + 7	93.9	55.3	63.2	61.4
Ref_Gen	5612	6272	5692	2612	16.2	12.9	11.6	18.7
Block_Ref_Buffer	2897				27.6			
MPM_Gen	639				5.3			
Orig_Pixel_Loader	15425				160.1			
Main Ctrl Logic	291				0.6			
CABAC_Coeff_Load_Ctr	2383				27.9			
CABAC Entropy Encoder	5155 + 2				26.6			
Total	201823 + 1253				2288			

Table 4.11 Comparison of FPGA implementations of H.265/HEVC intra encoders.

Architecture	Pastuszak [29]	Miyazawa [44]	Atapattu [45]	This work
FPGA	Arria II GX	N/A	Zyng ZC 706	Stratix V GX
Frequency (MHz)	100 / 200	N/A	140	120
Resolution	1080 P @ 60	1080 P @ 60	1080 P @ 30	1080 P @ 45
Support PU size	No 64×64 PU	N/A	N/A	All
Mode decision	RDO with bin counting	SAD	Early termination	RDO with highly-parallel table-based CABAC
RDO intra candidates	4-20 for each CU/PU/TU	N/A	35	1-7 for each CU/PU/TU
Intra quality losses	0.17dB	0.6 dB	N/A	0.11dB
BD-Rate increase	5.09%	N/A	13%	3.02%
Gate count	93K ALUTs + 481 DSPs	N/A	84K LUTs + 34 DSPs	201K ALUTs + 1253 DSPs
Memory (KB)	52	N/A	28	120

Table 4.10 reports resource utilization in the FPGA and ASIC implementations, including resource consumption details for four PEs and non-PE modules. Most of the logic resources are consumed by intra prediction, transform, quantization, inverse quantization, inverse transform, and rate estimation. Since the fully-parallel feature and inclusion of all PU and TU sizes, relatively more logic elements are utilized. The logic resources of transform, quantization, inverse quantization, and inverse transform heavily depend on computational throughput. In our architecture, the throughput of PE1 and PE3 are the slowest and highest, respectively. Hence, PE1 and PE3 consume the smallest and largest logic elements. Table 4.11 summarizes the performance comparison of existing FPGA implementations of H.265/HEVC intra encoders. The proposed design runs at 120MHz, supports all PU sizes, involves a smaller number of RDO intra candidates, and performs video encoding of 1080P @ 45 fps. In terms of video compression performance, the proposed FPGA implementation exhibits the minimum quality loss and BD-rate increase.

Table 4.12 Comparison of H.265/HEVC intra encoder hardware implementations.

Architecture	Zhu [28]	Tsai [32]	Huang [33]	Pastuszak [29]	This work
Function	Intra Encoder	Intra & Inter Encoder	Intra Encoder	Intra Encoder	Intra Encoder
Technology	TSMC 90nm	TSMC 28nm	SMIC 55nm	TSMC 90nm	TSMC 90nm
Frequency [MHz]	357	312	294	200/400	320
Supported PU Size	No 64×64 PU	No 4×4, 8×8 PUs	No 64×64 PU	No 64×64 PU	All
Supported TU Size	All	No 4×4 TU	No 32×32	All	All
RDO candidates	[4, 4, 4, 4]	N/A	[8, 4, 4, 2]	[12, 10, 4, 3]	[7, 5, 5, 4, 1]
Rate Estimator	Prediction Error Based Approximation	Table-based CFBAC	Table-based CABAC	Bin Counting	Highly-parallel Table-based CABAC
CABAC Entropy Encoder	Not included	Included	Not included	Included	Included
Resolution	1080P @ 44fps	8K @ 30fps	1080P @ 60fps	4K @ 30fps	4K @ 30fps
Quality Losses [BD-PSNR]	0.15dB (for 1080P)	0.8dB (for 8K)	N/A	0.17dB (for 1080P) 0.11dB (for 4K)	0.11dB (for 1080P) 0.10dB (for 4K)
Rate Increase [BD-Rate]	4.62% (for 1080P)	17% (for 8K)	5.86% (for 1080P)	5.09% for (1080P) 5.46% (for 4K)	3.02% (for 1080P) 4.37% (for 4K)
Gate Count	2269 K	8250 K	1572 K	1086 K	2288 K
Memory	N/A	7.14 MB	N/A	52 KB	120 KB
Power [mW]	217.9	708	194	273	236

Table 4.11 summarizes the performance comparison of the proposed H.265/HEVC intra encoder with existing state-of-the-art designs in the literature [28, 29, 32, and 33]. The proposed design is the only work that supports all PU and TU sizes, while the designs in [28, 29, 33] do not support 64×64 PUs that is highly desirable in ultra-high-definition 4K/8K videos, due to their excellent compression performance in smooth-textured videos. The design in [32] does not support 4×4 TUs, 4×4 and 8×8 PUs, yet, which are absolutely necessary to encode sharper and detail-rich videos. Regarding bit rate estimation performance, compared with the fast and simplified algorithms [28, 29], CABAC-based rate estimation provides the best accuracy with an overhead of more hardware cost. Our proposed highly-parallel CABAC rate estimator significantly increases the rate estimation throughput. The designs in [28, 32] are incomplete intra encoders, because CABAC entropy encoder is not included in their architectures. The alleviated data/timing dependency in our proposed fully-parallel architecture enables real-time video encoding up to 4K @ 30 fps. Thanks to the proposed hardware-oriented algorithm

adaptations in Section II, the rate increases (BD-Rate) in this work are 3.02% and 4.37% for 1080P and 4K video sequences, respectively. This BD-Rate is lower than the existing designs in [28, 32, and 33] with the same video resolution (1080P or 4K). The quality loss (BD-PSNR) in this work are 0.11dB and 0.10dB for 1080P and 4K video sequences, respectively. These results indicate the best coding efficiency over all existing designs in the literature. In this work, referring to the smallest two-input NAND gate, the total gate count is 2288K gate. The required memory size is 120KB, and the power consumption is 236 mW. These hardware cost, memory usage, and power consumption are comparable with the existing designs in Table 4.12.

In this work, in order to mitigate computational complexity, instead of using CABAC based rate estimation, chroma rate is estimated based on the number of non-zero coefficients after quantization stage. In future works, if CABAC based rate estimation is also adopted for chroma blocks, the BD-Rate in Table 4.12 is expected to shrink by another 1~2%. In this work, although four PEs are implemented to realize high parallelism, the overall throughput is still limited by the CABAC based rate estimators. As CABAC based rate estimation does not involve complex computations (such as multiplication), it is feasible to boost the clock frequency of this module for higher throughput.

4.5 CONCLUSION

In order to reduce computational complexity and mitigate data/timing dependency of H.265/HEVC intra encoding, this chapter presents efficient algorithm adaptations and a fully-parallel intra encoder hardware architecture. The proposed algorithm adaptations include PU chroma and luma mode preselection, modified CU mode decision, and simplified table-based CABAC rate estimation. Compared with the HM-15.0 software, the proposed algorithm adaptations lead to a reduction of 27% in computational workload, while the average BD-Rate

and BD-PSNR are 4.39% and -0.21dB, respectively. This BD-Rate is lower than the existing designs in literature with the same video resolution of 1080P or 4K. The proposed hardware architecture includes four independent parallel prediction engines. In order to achieve high accuracy, reliability, and throughput of rate estimation, multiple CABAC rate estimate instances are implemented, where 16 syntax elements are classified into five independent groups for parallel processing. The proposed intra encoder considers all CU/PU/TU sizes and 35 prediction modes, and is capable of performing real-time video coding up to 30 fps of 4K using TSMC 90nm process. This fully-parallel architecture is very promising to support even higher UHD video encoding with further modification, such as boosting the operating frequency of CABAC rate estimate instances.

CHAPTER 5

PERFORMANCE ENHANCED INTRA ENCODER

In this chapter, we propose a performance-enhanced intra encoder based on the proposed design in chapter 4. Particularly, we propose to include the derived luma mode that has been excluded to reduce mode dependency in our previous design, while still adopting other algorithm adaptations. The exclusion of derived luma mode has been observed resulting in significant BD-Rate increase for some test sequences, while the inclusion of the derived luma mode can drastically improve the coding efficiency of the worst cases. However, the inclusion of derived luma mode will bring mode dependency to intra prediction between chroma and luma blocks, resulting in a throughput decrease of intra prediction. In order to mitigate its influence, we propose several hardware strategies to improve the overall throughput of intra encoder. Compared with HM-15.0 reference software, the proposed algorithms reduce the computation workload by 22%, while the average BD-Rate and BD-PSNR for classes A-F and 4K are 2.79% and -0.13dB, respectively. This BD-Rate is lower than existing designs in literature with the same video resolution. With proper hardware modifications, the proposed intra encoder with TSMC 90nm technology can operate at 320 MHz with a hardware gate count of 2186K and power consumption of 290mW. Experimental results show our design supports real-time encoding of 4K videos at 30 fps.

5.1 INTRODUCTION

The enhanced coding features of H.265/HEVC bring more efficient compression than existing standards, however, the resultant computational complexity in RDO cannot be ignored. The challenges of implementing H.265/HEVC intra encoders have been discussed previously. To address these challenges, we have proposed hardware-oriented algorithms to reduce the

computational complexity, meanwhile, hardware architecture has been designed to validate the efficiency of proposed algorithms. In chapter 3, a highly-parallel table-based CABAC rate estimator has been proposed to accelerate rate prediction for the RDO process, while retaining excellent rate prediction accuracy. Along with other complexity reduction algorithms, we proposed a highly efficient H.265/HEVC intra encoder, which possesses a high compression efficiency with a 4.79% BD-Rate increase and 0.21dB BD-PSNR decrease on average in chapter 4. Hardware implementation with TSMC 90nm technology shows the proposed intra encoder is capable of processing 4K real-time videos at 30fps.

Although the overall compression performance of the proposed intra encoder in chapter 4 is superior to the state-of-the-art designs, we notice that relatively high BD-Rate increases are shown for some video sequences, such as Basketball Drill (12.12%), Race Horses (6.12%), and Slide Show (6.72%) [47]. Our initial investigation reveals that a remarkable increase of BD-Rate (*i.e.*, Basketball Drill (9.01%), Race Horses (4.00%), and Slide Show (3.85%)) occurs when applying the proposed PU chroma mode preselection algorithm to the proposed intra encoder in chapter 4. This observation inspires us to further improve the compression algorithms for HEVC intra encoders.

In this chapter, we firstly investigate the influence of the derived luma mode in chroma prediction. Experiments are conducted to perform intra encoding with derived luma mode in chroma prediction. Results are compared with that of proposed intra encoder in chapter 4. It turns out that the intra encoder with derived luma mode improves 1.6% over existing algorithms in BD-Rate on average. Particularly, in contrast to the intra encoder in chapter 4, the worst case of BD-Rate from Basketball Drill (12.12%) shifts to Slide Show (5.71%) in this design. It is certain that the inclusion of derived luma mode in PU chroma prediction brings back the mode

dependency between luma and chroma blocks, thereby reducing the overall compression throughput. In order to alleviate the resulting degradation in throughput, a modified hardware architecture of intra encoder is proposed in this chapter. Even if the basic system architecture adopts fully-parallel prediction engines (PEs), we propose new timing diagrams to properly allocate the processing orders for prediction and RDO processes in PEs. We also propose a more balanced prediction block schedule to mitigate the throughput degradation due to the inclusion of derived luma mode in PU chroma prediction. Moreover, we propose to double the operation frequency of context-adaptive binary arithmetic coding (CABAC) rate estimators to achieve an overall encoding throughput of 4K @ 30fps. In addition, asynchronous FIFOs are proposed to ensure correct timing when data/signals pass through two clock frequency domains. We discuss and provide solutions for the number and depth of asynchronous FIFOs for each PE. This hardware architecture is described in Verilog and synthesized in an FPGA and an ASIC using TSMC 90nm technology. This ASIC implementation uses 127K on-chip memory and 2186K logic gates, consumes 290mW, supports 320MHz clock frequency, and sustains 4K video sequences at 30 fps. Compared to the state-of-the-art designs [28, 29, 32, 33, and 48], it provides the best compression performance with a little overhead on memory and power consumption.

The rest of this chapter is organized as follows. The proposed high-performance adaptation is described in section 5.2. Section 5.3 introduces the hardware architecture and timing diagrams. System implementation results are presented and compared in section 5.4. Finally, section 5.5 concludes the chapter.

5.2 PROPOSED HIGH-PERFORMANCE ALGORITHM ADAPTATION

In the beginning, we investigate the reason why video compression performance is relatively higher for certain videos (*e.g.*, Basketball Drill) when using the previously proposed

algorithm. In chapter 4, the PU chroma mode preselection algorithm consists of two independent algorithms: the derived luma mode removal, and simplified chroma rate estimation. The derived luma mode removal aims to eliminate the mode dependency of intra prediction between a luma block and its chroma blocks. In HEVC, chroma prediction contains four fixed modes (*i.e.*, DC, Planar, Vertical, and Horizontal) and one derived mode. As the derived mode is determined by the luma RDO process, the chroma prediction cannot start before the completion of a luma RDO. To get rid of this inherent mode dependency, the algorithm in chapter 4 removed this derived luma mode in chroma prediction. Regarding the simplified chroma rate estimation, due to the serial syntax processing of CABAC and the high throughput requirement of UHD video coding, it is not necessary to implement CABAC-based rate estimation for chroma PUs. Moreover, since humans are more sensitive to luminance than chrominance, it is reasonable to approximate the rate estimation computation of chroma PUs by counting the non-zero quantization coefficients. This simplified chroma rate estimation has the benefits of low complexity, high throughput, efficient area, and low power consumption.

Table 5.1 Breakdown results of BD-Rate increase in M1 algorithm.

		Derived luma mode removal	**Simplified chroma rate estimation**	**PU chroma mode preselection**
Class	**Sequences**	**BD-Rate [%]**	**BD-Rate [%]**	**BD-Rate [%]**
C	Basketball Drill	8.03	1.16	9.01
D	Basketball Pass	3.22	1.13	4.16
D	Race Horses	2.92	1.14	4.00
F	Slide Show	3.05	1.96	3.85

Table 5.1 shows the breakdown BD-Rate increase caused by the PU chroma mode preselection algorithm. It shows at least 80% of the BD-Rate increase is attributed to the derived luma mode removal algorithm. The derived luma mode is one of five intra prediction modes for chroma prediction, while the other four intra prediction modes are fixed modes (i.e., Planar, DC,

Vertical, and Horizontal). This inherent dependence of the prediction mode between luma and chroma PUs results in data and timing dependencies. In our previous design, this derived luma mode is removed to eliminate data and timing dependencies. However, according to Table 5.1, the derived luma mode plays a key role in video compression performance, especially for the worst-case video sequences. Therefore, we propose to include the derived luma mode in the PU chroma mode prediction algorithm.

To reduce the computational complexity and improve video compression efficiency, especially for the worst-case videos, three algorithm adaptations are proposed in this work. First, simplified chroma rate estimation is used in the chroma RDO process, where the chroma rate is estimated by counting the number of non-zero quantized coefficients. All five chroma modes, including one derived luma mode and four fixed modes are supported in this algorithm. Second, mode preselection in luma prediction is proposed. In order to maintain low computational complexity for luma mode preselection, the most probable modes (MPMs), regular modes, and Hadamard mode are selected. Moreover, in order to balance the throughput of various PUs, different numbers of RDO candidates are utilized for different PU sizes. Third, a fast RD cost estimation for CUs is proposed. To reduce the redundant computation and hardware complexity, instead of re-calculating the CU rate and distortion, the RD cost of CUs is estimated based on the luma and chroma RDO results. In addition, table-based CABAC rate estimation of luma PUs is modified slightly to reduce the complexity of hardware implementation. Particularly, a syntax element "*split_cu_flag*" is omitted from luma rate estimation, due to the following considerations. From an algorithm point of view, this flag only involves in the comparison between CU and sub-CUs. This syntax element is not involved in any RD comparison among various prediction modes of a given CU. Removal of this syntax element from rate estimation

reduces the computational complexity and simplifies hardware circuit implementation. Compared with the HM-15.0 reference software, the proposed algorithm adaptions reduce the computational workload by 22%. Our algorithm adaptations remain low data and timing dependencies that indicate the potential for high-throughput hardware implementation.

The proposed algorithm adaptations have been implemented in the HM-15.0 reference software. Algorithm simulations are carried out to evaluate video compression performance. All simulation results are obtained with residual-quad-tree (RQT) disabled. Table 5.2 compares the losses in the compression efficiency between the existing designs [28, 29, 33, and 47] and this design. It is clear that the inclusion of derived luma mode in the PU chroma mode prediction significantly improves the video compression efficiency. To be specific, the average BD-Rate in this work is 1.97% and 3.79% for 1080P and 4K video sequences, respectively. These BD-Rate values are lower than the existing designs in [28, 29, 33, and 47] with the same video resolution (1080P or 4K). The average rate increase (BD-Rate) and quality loss (BD-PSNR) over all the video sequences (i.e., classes A-F and 4K) are 2.79% and 0.13dB, respectively. If the video sequences are classes A-E except for Party Scene, Race Horses, and Four People, the average rate increase (BD-Rate) and quality loss (BD-PSNR) are 2.41% and 0.11dB, respectively, which are far better than the results in [28, 29, 33, and 47]. More importantly, the proposed algorithm also greatly improves the worst-case compression performance for classes A-F and 4K. Particularly, in contrast to our previous design, the worst case of BD-Rate from Basketball Drill (12.12%) shifts to Slide Show (5.71%) in this work. The worst-case BD-Rate in this work is also better than [28, 29, 33, and 47]. Compared with the HM-15.0 reference software, the reduction of computational workloads in chapter 4 is 27%, while it is 22% in this work. Therefore, this work significantly improves video compression efficiency with a little overhead on computational

efforts, especially for the worst-case video sequences. To realize the proposed algorithm adaptations, a new hardware architecture needs to be developed with high parallelism and throughput.

Table 5.2 Efficiency comparison between different intra encoding algorithms.

Class	Sequences	Zhu [28]		Huang [33]		Pastuszak [29]		Zhang [47]		This work	
		BD-Rate [%]	BD-PSNR [dB]	BD-Rate [%]	BD-PSNR [dB]	BD-Rate [%]	BD-PSNR [dB]	BD-Rate [%]	BD-PSNR [dB]	BD-Rate [%]	BD-PSNR [dB]
A	People On Street	4.61	-0.21	3.1	N/A	5.18	-0.25	3.31	-0.17	1.97	-0.10
A	Traffic	4.34	-0.21	4.0	N/A	4.73	-0.22	4.49	-0.22	3.79	-0.18
B	Park Scene	3.39	-0.11	3.2	N/A	3.97	-0.15	2.04	-0.09	1.75	-0.07
B	Kimono	4.39	-0.12	11.2	N/A	3.42	-0.11	2.92	-0.09	2.72	-0.09
B	Basketball Drive	6.73	-0.17	7.7	N/A	7.39	-0.20	4.33	-0.13	1.53	-0.04
B	BQ Terrace	4.32	-0.19	3.1	N/A	4.99	-0.20	1.95	-0.09	0.94	-0.04
B	Cactus	4.28	-0.14	4.1	N/A	5.70	-0.18	3.87	-0.14	2.90	-0.09
C	Basketball Drill	4.63	-0.21	4.2	N/A	8.91	-0.39	12.12	-0.59	4.88	-0.22
C	BQ Mall	4.15	-0.20	3.8	N/A	6.23	-0.30	3.30	-0.17	1.71	-0.08
C	Party Scene	N/A	N/A	1.9	N/A	5.39	-0.34	2.76	-0.21	1.82	-0.11
C	Race Horses	3.38	-0.19	2.8	N/A	5.74	-0.29	4.97	-0.29	2.81	-0.14
D	Basketball Pass	4.80	-0.24	4.6	N/A	7.28	-0.39	6.02	-0.37	3.42	-0.19
D	Blowing Bubbles	3.44	-0.19	2.2	N/A	5.91	-0.35	3.63	-0.27	2.29	-0.14
D	BQ Square	1.97	-0.15	1.3	N/A	6.41	-0.42	2.37	-0.18	1.46	-0.10
D	Race Horses	N/A	N/A	3.2	N/A	6.78	-0.36	6.12	-0.40	3.78	-0.21
E	Kristen And Sara	5.86	-0.24	6.3	N/A	7.88	-0.36	6.28	-0.26	2.53	-0.12
E	Four People	N/A	N/A	4.1	N/A	5.79	-0.29	3.33	-0.16	1.78	-0.09
E	Johnny	5.15	-0.21	9.8	N/A	8.03	-0.30	5.31	-0.19	1.44	-0.06
F	Slide Editing	N/A	N/A	2.0	N/A	N/A	N/A	2.48	-0.36	2.20	-0.28
F	Slide Show	N/A	N/A	6.4	N/A	N/A	N/A	6.72	-0.59	5.71	-0.48
F	China Speed	N/A	N/A	2.7	N/A	N/A	N/A	4.12	-0.37	3.08	-0.25
4K	Beauty	N/A	N/A	N/A	N/A	N/A	N/A	5.06	-0.08	4.13	-0.05
4K	Bosphorus	N/A	N/A	N/A	N/A	6.03	-0.16	6.19	-0.16	5.29	-0.13
4K	Honey Bee	N/A	N/A	N/A	N/A	5.35	-0.08	2.31	-0.04	1.81	-0.02
4K	Jockey	N/A	N/A	N/A	N/A	7.31	-0.10	4.54	-0.08	3.46	-0.04
4K	Ready Steady Go	N/A	N/A	N/A	N/A	5.37	-0.16	4.84	-0.11	3.63	-0.10
4K	Shake & Dry	N/A	N/A	N/A	N/A	2.58	-0.06	1.84	-0.04	1.19	-0.02
4K	YachtRide	N/A	N/A	N/A	N/A	6.11	-0.13	5.79	-0.16	4.23	-0.13
Avg class B (1080P HD)		4.62	-0.15	5.86	N/A	5.09	-0.17	3.02	-0.11	1.97	-0.07
Average class 4K except for Beauty		N/A		N/A	N/A	5.46	-0.12	4.25	-0.10	3.27	-0.07
Average class 4K		N/A		N/A	N/A	N/A		4.99	-0.13	3.79	-0.09
Average 1080P HD and 4K except for Beauty		N/A		N/A	N/A	5.29	-0.14	3.69	-0.10	2.68	-0.07
Average value for classes A-E except for Party Scene, Race Horses, and Four People		4.36	-0.19	4.76	N/A	6.12	-0.27	4.46	-0.22	2.41	-0.11
Average for classes A-F and 4K		N/A		N/A	N/A	N/A		4.39	-0.21	2.79	-0.13
Worst-Case of classes A-F and 4K		**6.73**	**-0.17**	**11.2**	N/A	**8.91**	**-0.39**	**12.12**	**-0.59**	**5.71**	**-0.48**

5.3 HARDWARE ARCHITECTURE AND TIMING DIAGRAM

5.3.1 Proposed Hardware Architecture

The proposed high-throughput hardware architecture is developed based on the fully-parallel intra encoder architecture in chapter 4. As mentioned earlier, when the derived luma

mode is added to the chroma prediction, the intrinsic mode dependency introduces data and timing latency during the chroma rate-distortion optimization process. As a result, the inclusion of derived luma mode in PU chroma prediction reduces the throughput of video comparison. To mitigate the degradation in throughput, we propose to double the clock frequency of CABAC rate estimation modules in the hardware architecture of HEVC intra encoders.

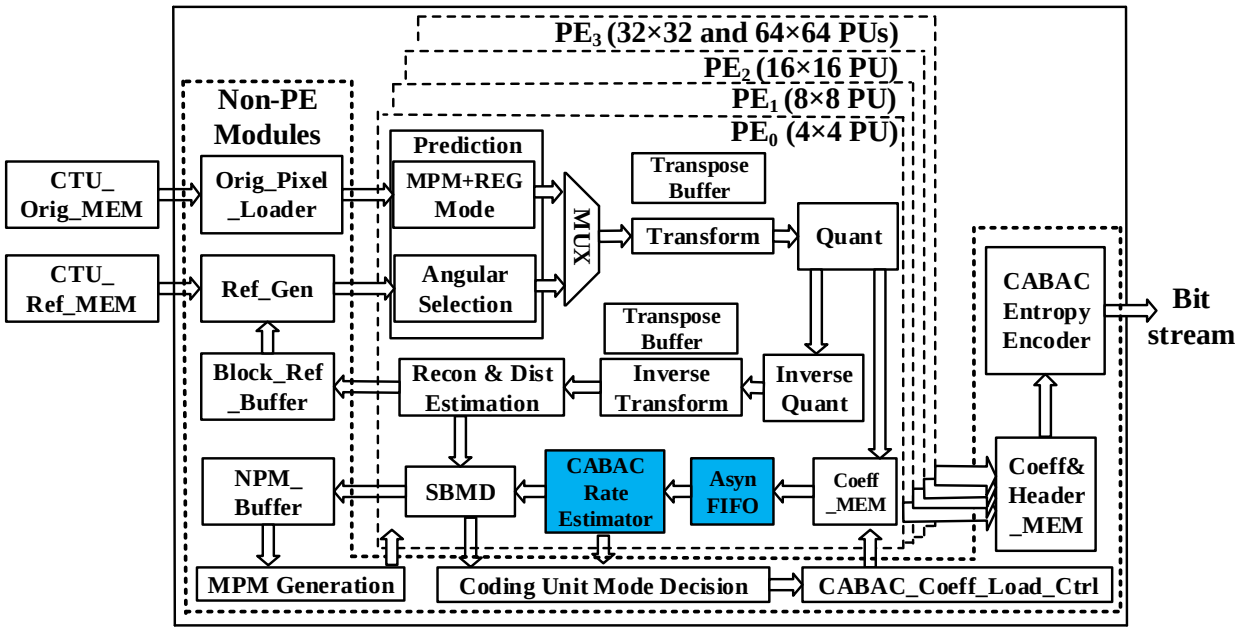


Fig. 5.1. Hardware architecture overview of proposed HEVC intra encoder.

Fig. 5.1 shows a block diagram of the proposed hardware architecture. All functional modules are divided into two groups: prediction engine (PE) modules and non-prediction engine (Non-PE) modules. The details of design considerations and implementations of each block are found in [47]. Four parallel PEs perform computational tasks in intra mode prediction, transform, quantization, inverse quantization, inverse transform, reconstruction, CABAC based rate estimation, distortion estimation, and simplified block mode decision (SBMD). Each PE is dedicated to a particular PU size, except for PE3 that is shared by 32×32 and 64×64 PUs for area efficiency. In this new hardware architecture, one PE that performs chroma prediction has to run intra prediction for the derived luma mode. For example, PE0 runs chroma prediction for two

additional luma modes derived from 4×4 and 8×8 PUs. PE1 performs chroma prediction for the luma modes derived from 16×16 PUs. PE2 performs chroma prediction for two additional luma modes derived from 32×32 and 64×64 PUs. The remaining Non-PE modules perform fewer computational tasks, such as reference pixel storage and preparation, adjacent prediction mode storage and fetch, MPM generation, original pixel loading, and CABAC entropy coding. The inclusion of derived luma mode in chroma prediction requires a certain amount of on-chip storage memory. The discussion of additional memory usage will be described in section 5.3.4.

5.3.2 Proposed Timing Diagram and Balanced Prediction

Let us analyze and illustrate the timing diagram of $PE_{0\text{-}2}$ and their specific data/timing dependencies. Assuming the video format is YCbCr 4:2:0, the chroma block size of each CU is a quarter of its luma block size. For example, a 16×16 CU has one 16×16 luma prediction block and two 8×8 chroma prediction blocks. The only exception is 8×8 CUs with 4×4 partition, which has the same size of luma and chroma blocks. In order to maximize intra prediction throughput, four PEs will process different PU sizes simultaneously. Due to the derived luma mode dependency between PEs, extra latency occurs in chroma prediction.

Fig. 5.2 plots an illustrative timing diagram of $PE_{0\text{-}2}$, which includes the latencies due to the inclusion of derived luma mode. As each prediction mode in PE_0, PE_1, and PE_2 takes 1, 8, and 16 clock cycles, respectively, the number of clock cycles required to predict 4×4, 8×8, and 16×16 PUs is marked in Fig. 5.2. In addition, the number of clock cycles required for an RDO process is determined by the quantization parameter (QP) and CABAC rate estimation. Due to the context-adaptive feature of CABAC rate estimation, the time required for CABAC rate estimation varies with prediction blocks and QPs. Therefore, we don't specify the number of clock cycles for RDO processes in Fig. 5.2. As shown in Fig. 5.2, for each 8×8 CU, two

partitions (i.e., four 4×4 PUs or one 8×8 PU) will be explored and evaluated in PE_0 and PE_1, respectively. Besides the four regular chroma modes (i.e., Planar, DC, Vertical, Horizontal), two additional chroma modes are added in PE_0. One mode (J, K) is derived from 4×4 PUs, while the other mode (M, N) is derived from 8×8 PUs. Since the mode (J, K) has already been derived from the RDO process of the first 4×4 luma PU, no extra latency occurs in PE_0. However, the prediction of mode (M, N) cannot start until the luma mode of 8×8 PUs is derived from PE_1. If the RDO process of 8×8 luma PUs is complete before intra prediction of mode (M, N), the derived luma mode dependency does not result in a significant drop in throughput. Otherwise, intra prediction of mode (M, N) will be postponed until the completion of the RDO process of 8×8 luma PUs. In Fig. 5.2, luma and chroma prediction blocks of 16×16 CUs are processed in PE_2 and PE_1, respectively. The four regular chroma modes are appended to four 8×8 PUs, denoted as UV0-3. Intra prediction of the derived luma mode of 16×16 PUs is denoted as UV4. If the luma mode of a 16×16 PU has already been determined, prediction of the derived luma mode of this 16×16 PU will start immediately after UV3. Otherwise, intra prediction of the derived luma mode will be postponed until the completion of the RDO process of 16×16 luma PUs. In addition to the prediction of 16×16 luma PUs, PE_2 needs to perform chroma prediction for 32×32 and 64×64 PUs.

The original timing diagram of PU chroma prediction is unbalanced as illustrated in Fig. 5.3, where four regular chroma modes are evenly distributed to four 16×16 luma PUs for intra prediction, while the prediction of derived luma mode is appended to the fourth 16×16 luma PU. Since there are two chroma components (chroma U, chroma V) for each mode, the overall 16×16 prediction block numbers of four 16×16 PUs are 9, 9, 9, and 13, respectively. Obviously, due to the inclusion of derived luma mode, the fourth 16×16 PU needs to process four more chroma

prediction blocks than other three 16×16 PUs. This unbalanced prediction block scheme will definitely cause a throughput degradation, when the RDO process of current 16×16 CU has completed, but PE_2 still performs chroma prediction for 32×32 or 64×64 CUs. Therefore, a balanced chroma prediction scheme is proposed to mitigate prediction burden of the last 16×16 PU. In the proposed scheme in Fig. 5.3, two regular chroma modes of 32×32 CUs will be processed in the third 16×16 PUs, along with one regular chroma mode of 64×64 CUs. Thus, the total prediction block numbers for corresponding PUs are 9, 9, 11, and 11, respectively. This schedule rebalance improves the overall processing throughput with a negligible overhead of hardware cost.

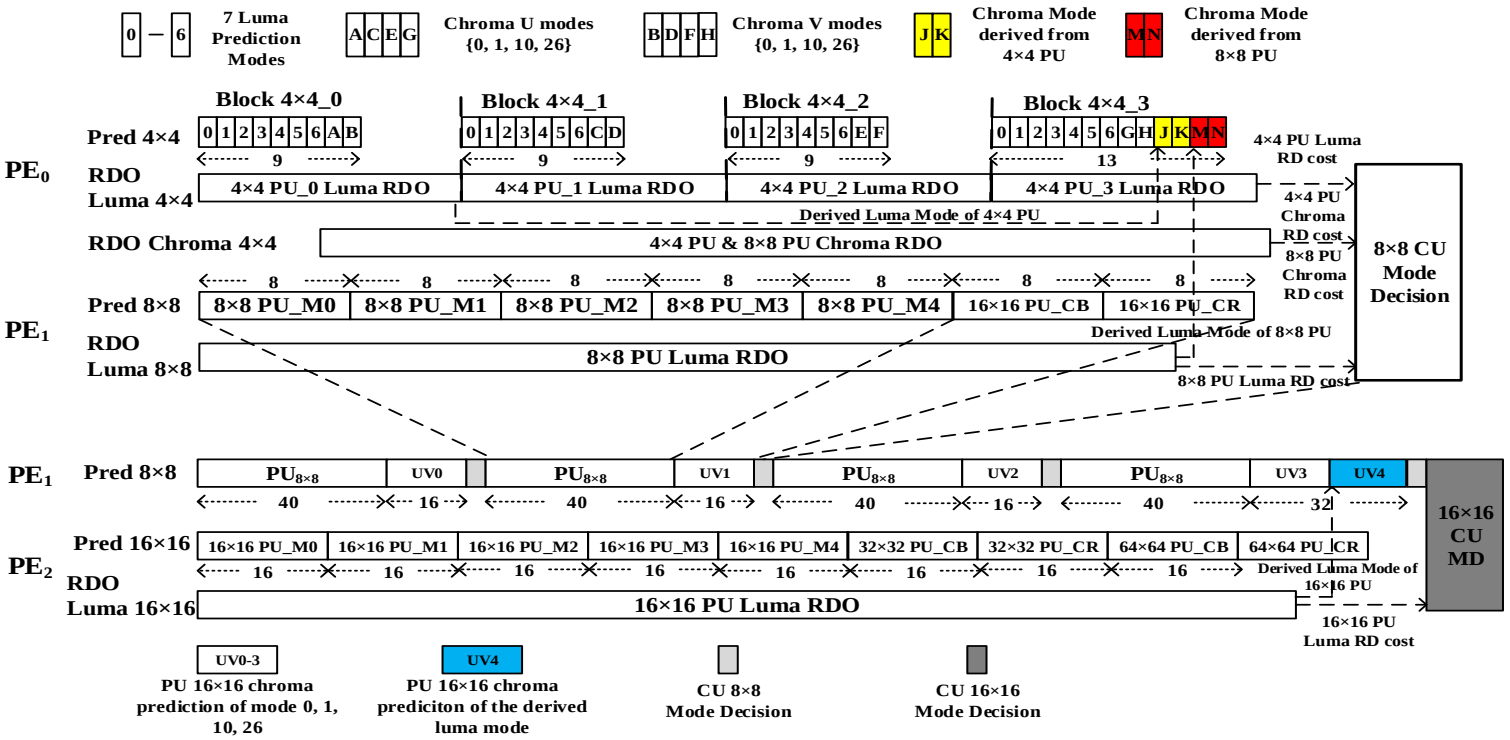


Fig. 5.2. Timing diagram of $PE_{0\text{-}2}$ and their data/timing dependency.

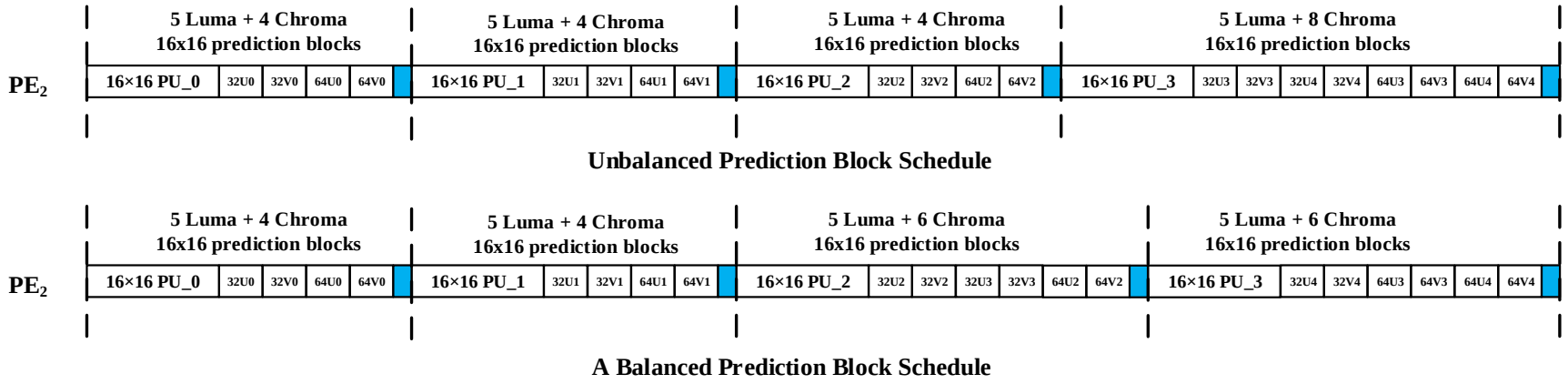


Fig. 5.3. The unbalanced and balanced block schduling schemes.

5.3.3 Proposed Double-Clock Rate Estimation

Since the table-based CABAC algorithm is adopted in luma PU rate estimation, the overall system throughput is severely affected by the rate estimation throughput. Although the computation in the CABAC rate estimators is conceptually similar to the CABAC entropy encoder, such as syntax binarization, context-adaptive modeling. The difference is that the CABAC entropy encoder outputs an encoded bit-stream, while the CABAC rate estimator outputs a bitrate estimate. The details of the hardware implementation of table-based CABAC rate estimators are described in [48]. To mitigate the throughput degradation due to the derived luma mode in PU chroma prediction, CABAC rate estimators are proposed to operate at a double clock rate. This is because CABAC rate estimators are neither computationally-intensive nor logic-complex, they can operate at a much higher clock rate (*e.g.*, 640 MHz with TSMC 90nm process), while remaining blocks in the intra encoder operate at a lower clock rate (*e.g.*, 320MHz with TSMC 90nm process). In this work, 13 rate estimate instances are employed to satisfy the performance requirements of UHD video compression. Specifically,

7, 2, 2, 1, and 1 rate estimate instances are allocated to intra prediction of 4×4, 8×8, 16×16, 32×32, and 64×64 PUs, respectively.

In this dual clock rate hardware architecture, the correctness of data and signal transition across the clock domains is required. In this work, a handshake synchronizer is used to ensure proper transmission of control signals across the boundaries of two clock domains. In addition, an asynchronous first-in-first-out (AFIFO) register is used to handle different reading and writing frequencies of quantized coefficients. As shown in Fig. 5.1, an AFIFO block is located between a CABAC rate estimator and Coeff_MEM inside each PE. Table 5.3 lists the detailed information of AFIFOs in each PE. The width of all AFIFOs is 256, and the depth varies with PE. In PE_0, an AFIFO of depth 1 is implemented for all 7 rate estimate instances. Because the writing frequency is much slower than the reading frequency, if a 4×4 block of a rate estimate instance is available, all 7 rate estimate instances immediately read the AFIFO. However, in the PE_1~PE_3, the AFIFO depth is set to 4. For 8×8 and 16×16 PUs, two AFIFOs have been instantiated and each AFIFO corresponds to one rate estimate instance.

Table 5.3 Asynchronous FIFO number and depth in each PE.

PE	**Number of RDO candidates**	**AFIFO**	
		Number	**Depth**
PE_0	7	1	1
PE_1	2	2	4
PE_2	2	2	4
PE_3	1 for 32×32 PUs, 1 for 64×64 PUs	2	4

5.3.4 Memory Usage

Since an additional chroma mode is added for each chroma PU, a little more on-chip memory is required in this proposed hardware architecture. Since the on-chip memory is shared for both luma and chroma modes in original and predicted pixels, the on-chip memory size for original pixels and predicted pixels remains the same as in [47]. The on-chip quantization

memory adds 43,008 bits, and the on-chip reconstruction memory adds 21,504 bits. Table 5.4 summarizes the on-chip memory usage of all PE and non-PE modules. Compared with our previous work [47], where the total on-chip memory usage is 120KB, this proposed intra encoder hardware architecture needs a total amount of 127KB on-chip memory. This result shows that the inclusion of derived luma mode needs 7KB on-chip memory, so our proposed algorithm adaptations do not result in a significant increase in memory usage.

Table 5.4 Memory usage in the proposed intra encoder (unit: bit).

	ORIG_MEM	PRED_MEM	QUANT_MEM	RECON_MEM	Total(PE)
PE_0	Register	Register	Register	Register	None
PE_1	2×24×32	2×40×32	2×120×64	1×120×64	27,136
PE_2	4×48×32	4×80×32	4×880×64	2×880×64	354,304
PE_3	8×32×32	8×160×32	8×256×64	4×256×64	245,760
Total (PE)	15,872	53,760	371,712	185,856	627,200
Non-PE	Coeff&Header_MEM				256,960
	CTU Orig_MEM				98,304
	CTU Ref_MEM				62,464
Total	1,044,928				

5.4 EXPERIMENTAL IMPLEMENTATION AND RESULTS.

The proposed highly-parallel hardware architecture of HEVC intra encoder has been described in Verilog and synthesized by field programmable gate array (FPGA) and application specific integrated circuit (ASIC). The FPGA implementation is based on Altera Stratix V GX, and the ASIC implementation uses TSMC 90nm technology. The maximum operating clock frequencies for the FPGA and ASIC implementations are 120MHz and 320MHz, respectively.

Table 5.5 Resource comparison between FPGA and ASIC implementations.

Module	Stratix V GX (# of ALUT + # of DSP)				TSMC 90nm (K gate)			
	PE_0	PE_1	PE_2	PE_3	PE_0	PE_1	PE_2	PE_3
Intra Prediction	7268 + 64	12616 + 64	19666 + 82	25660 + 106	61.1	97.1	124.4	185.9
Trans	2568 + 56	1443+ 24	3133 + 50	10803 + 224	33.8	24.1	65.8	244.0
Quant	1127 + 16	582 + 8	1309 + 16	2497+ 32	28.2	13.9	29.0	57.6
iQuant	368 + 16	184+ 8	368 + 16	704 + 32	24.9	12.3	24.3	47.4
iTrans	3820 + 56	1782 + 24	3542 + 52	11161 + 222	38.2	26.7	61.9	236.0
Reconstruction	320	160	320	640	1.8	0.9	1.7	3.4
Distortion Calculation	401 + 12	197 + 6	410 + 12	817+ 24	7.2	3.4	6.9	13.5
Rate Est & SBMD	11183 + 8	6043 + 3	5924 + 4	5461 + 2	84.4	69.4	76.4	68.6
Ref_Gen	5616	6281	5574	3022	15.4	12.7	12.3	19.5
Block_Ref_Buffer	2267				24.7			
MPM_Gen	622				5.4			
Orig_Pixel_Loader	16155				165.8			
Main Ctrl Logic	305				1.1			
CABAC_Coeff_Load_Ctr	2683				34.2			
CABAC Entropy Encoder	5119 + 2				32.6			
Total	195883 + 1244				2186.4			

Table 5.6 Comparison of H.265/HEVC intra encoder hardware implementations.

Architecture	Zhu [28]	Tsai [32]	Huang [33]	Pastuszak [29]	Zhang [47]	This work
Technology	TSMC 90nm	TSMC 28nm	SMIC 55nm	TSMC 90nm	TSMC 90nm	TSMC 90nm
Frequency [MHz]	357	312	294	200/400	320	320/640
Supported PU Size	No 64×64 PU	No 4×4, 8×8 PUs	No 64×64 PU	No 64×64 PU	All	All
Supported TU Size	All	No 4×4 TU	No 32×32	All	All	All
RDO luma candidates	[4, 4, 4, 4]	N/A	[8, 4, 4, 2]	[12, 10, 4, 3]	[7, 5, 5, 4, 1]	[7, 5, 5, 4, 1]
RDO chroma candidates	N/A	N/A	1	5	4 for all PUs	5 for all PUs
Rate Estimator	Prediction Error Based Approximation	Table-based CFBAC	Table-based CABAC	Bin Counting	Highly-parallel Table-based CABAC	Highly-parallel Table-based CABAC
Resolution	1080P @ 44fps	8K @ 30fps	1080P @ 60fps	4K @ 30fps	4K @ 30fps	4K @ 30fps
Quality Losses [BD-PSNR]	0.15dB (for 1080P)	0.8dB (for 8K)	N/A	0.17dB (for 1080P) 0.11dB (for 4K)	0.11dB (for 1080P) 0.13dB (for 4K)	0.07dB (for 1080P) 0.09dB (for 4K)
Rate Increase [BD-Rate]	4.62% (for 1080P)	17% (for 8K)	5.86% (for 1080P)	5.09% for (1080P) 5.46% (for 4K)	3.02% (for 1080P) 4.99% (for 4K)	1.97% (for 1080P) 3.79% (for 4K)
Worst-case Quality Losses [BD-PSNR]	0.17dB	N/A	N/A	0.39dB	0.59dB	0.48dB
Worst-case Rate Increase [BD-Rate]	6.73%	N/A	11.2%	8.91%	12.12%	5.71%
Gate Count + Memory	2269 K + N/A	8250 K + 7.14 MB	1572 K + N/A	1086 K + 52 KB	2288 K + 120 KB	2186 K + 127 KB
Power [mW]	217.9	708	194	273	236	290

Table 5.5 reports the resource unitization in the FPGA implementation, including resource consumption details for each module. The clock gating technique is utilized for low power purpose during the ASIC synthesis process. Table 5.6 summarizes the hardware implementation and performance comparison of this work with existing state-of-the-art designs

[28, 29, 32, 33, and 47]. This work supports all PU and TU sizes, as well as five RDO chroma candidates. The maximum operating clock frequencies for CABAC rate estimators and other modules are 640MHz and 320MHz, respectively. Under the same encoding throughput, this proposed work achieves the lowest quality loss (BD-PSNR) and rate increase (BD-Rate) over the designs in [28, 29, 32, 33, and 47]. These results indicate the best video compression efficiency over all existing designs in the literature. We can see that the throughput degradation due to the inclusion of derived luma mode is successfully compensated by doubling clock frequency in CABAC rate estimators. Regarding the worst-case rate increase and quality losses, Table 5.6 shows that the proposed design achieves the best compression performance for the worst-case video sequences in classes A-F and 4K. Particularly, in contrast to [47], the worst-case BD-Rate is improved from 12.12% to 5.71%.

Referring to the smallest two-input NAND gate, the total gate count is 2186K. Compared with [47], the logic gate count of this design is slightly lower. This slight reduction is explained below. First, in this work, since the derived luma mode is implemented in chroma prediction, extra logic gates are required. Second, in this work, the implementation of dual-clock rate CABAC rate estimation and asynchronous FIFOs increase logic gate consumption. Third, in this work, a great number of redundant logics are removed from the entire design, which significantly reduces the number of logic gates. For example, compared with Table 4.10 in [47], Table 5.5 shows that the removal of redundant logic in transforms and inverse transforms in PE_0 and PE_3 reduces around 80K logic gates. Due to the above three reasons, the total logic count of this work is slightly lower than the design in [47]. Compared with [47], the power consumption of this work increases by 54mW, this is mainly attributed to the higher operation frequency of CABAC rate estimators. Considering the fact that power consumption is a design metric that is relatively

less critical than video compression performance and throughput, it seems like a good trade-off to spend a little bit more power in significantly improving compression quality, especially for the worst-case videos. In practice, if the required video encoding throughput is not very high (*e.g.*, 4K@20fps, 1080P@60fps), there is no need to double the clock frequency for rate estimators. Thus, our entire intra encoder ASIC can be configured to run at 320MHz. In this way, our proposed design will achieve a similar level of power consumption as [47], while still performing much better video compression performance than the designs in [28, 29, 32, 33, and 47].

5.5 CONCLUSION

This chapter presents efficient algorithm adaptations and a high-throughput HEVC intra encoder hardware architecture. We identify that the derived luma mode has a non-negligible effect in video compression efficiency. Therefore, we propose to include it in PU chroma prediction, along with simplified chroma rate estimation, PU luma mode preselection, modified CU mode decision, and simplified CABAC rate estimation. Compared with the HM-15.0 software, this work saves computational efforts by 22%, while the average BD-Rate and BD-PSNR for all test sequences are 2.79% and -0.13dB, respectively. More importantly, the proposed algorithm adaptations greatly improve the worst-case compression performance over existing designs in the literature. We also develop an efficient high-throughput hardware architecture. We propose new timing diagrams to properly allocate the processing orders for prediction and RDO processes in PEs. We also propose a more balanced prediction block schedule to mitigate the throughput degradation due to the inclusion of derived luma mode in PU chroma prediction. The entire design has been synthesized and implemented in an FPGA and an ASIC. The ASIC implementation in TSMC 90nm technology shows that it supports real-time

high-throughput UHD video encoding of 4K videos at 30 fps with a little overhead increase in on-chip memory and power consumption.

CHAPTER 6

DEEP LEARNING BASED MODE PRESELECTION

In recent years, the rapid development of machine learning approaches has been applied to a wide range of fields, such as autonomous driving, intelligent robotics, computer vision, speech analysis, object detection, and classification. Among various machine learning strategies, deep learning has been proved to be effective in some tasks, where traditional mathematical approaches do not perform these tasks well [49]. The superior ability to explore the potential characteristics of input data is the greatest advantage of deep learning over mathematical algorithms [50]. Advances in technologies such as integrated circuit (IC) design, IC fabrication, parallel and cloud computation, and communications have made it possible to conduct deep learning through extensive training data. Inspired by its superior performance to human experts, we propose to introduce a deep learning method to improve the performance of the proposed HEVC intra encoder. A neural network that contains only fully connected layers is employed to roughly select a mode for luma blocks. The proposed fully connected layer based neural network (FCLNN) is incorporated into a two-stage mode preselection scheme, which outputs one or two modes to an intra predictor. Experiments have shown that the proposed mode preselection scheme can achieve mode accuracy of approximately 85%.

6.1 INTRODUCTION

The concept of Machine Learning (ML) was proposed and interpreted by Arthur Samuel, Tom M. Mitchell, and Alan Turing in the middle 20th century. ML investigates algorithms and structures that enable computers to perform complex tasks manually. As one of the most important ML frameworks, artificial neural networks (ANNs) have recently attracted more and more interest. An ANN consists of artificial neurons (nodes) that mimic biological neurons in the

human brain. Multiple nodes can form a layer, and a neural network typically has more than one layer. Similar to a synapse, a connection between two nodes is defined to transfer signals from one layer to another. Therefore, there are no connections within a layer. Each neural node receives input data, processes the received data, and sends the result to the next layer. A common neural network includes an input layer for receiving external data, a hidden layer for processing input data from the previous layer and sending the results to the next layer, and an output layer for sending results out of the neural network.

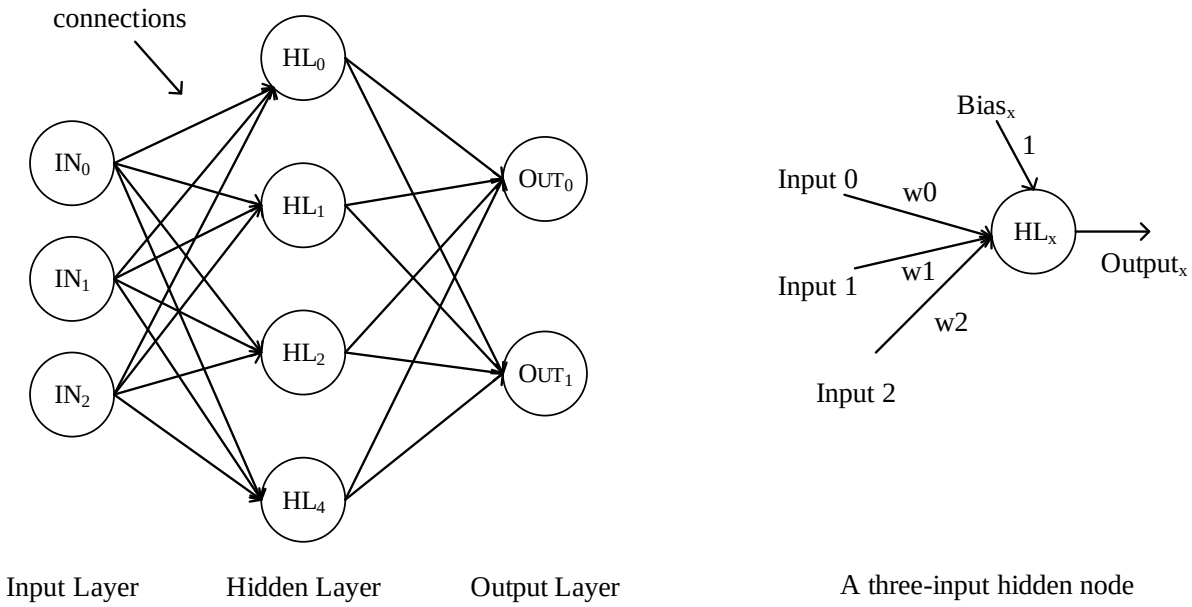


Fig. 6.1. Basic neural network structure (left) and neural node (right).

As shown in Fig. 6.1, this neural network contains three layers with a feedforward topology, where the data flow from input to output is strictly feedforward. Each hidden node receives input signals and produces the corresponding result using the following equation:

$$\text{Output}_x = \text{Func}\ \{\sum (\text{Input}_i * w_i) + \text{Bias}_x\} \qquad (6.1)$$

where *Func* is an activation function for determining the output of a neuron according to the weighted sum of input signals plus the bias term. The activation function is used to constrain the produced result of each node within a valid range, which makes sense for the subsequent

neurons. A number of activation functions have been proposed, such as Sigmoid, ReLu, Tanh, etc. ReLu (rectified linear unit) is the most widely used activation function in neural networks. Its idea is to convert a negative input to zero, while keeping a positive input unchanged.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{6.2}$$

As shown in (6.2), ReLu is a simple activation function, but it has several advantages. First, it is a nonlinear function, which means stacked layers can provide more possibilities. Thus, multiple layers can be used. Second, ReLu is a sparse function that disables insignificant nodes to save computation and energy. Compared to Sigmoid and Tanh, it is less computationally expensive and more suitable for large-scale networks.

In this dissertation, we propose a two-stage HEVC mode preselection scheme, which roughly selects intra prediction modes of each PU in a CTU. The first stage of the proposed mode preselection scheme determines the mode group for each PU, which includes prediction modes with similar prediction directions. According to different prediction angles, all 33 intra prediction modes are divided into 8 groups. So each mode group contains 4 or 5 prediction modes. In the second stage, a fully connected layer based neural network (FCLNN) is adopted to further refine the mode prediction results of the first stage. The proposed neural network is feedforward and utilizes ReLu as its activation function. Experiments are conducted to determine the numbers of neurons and layers.

The rest of this chapter is organized as follows. A literature review of machine learning approaches in video compression is presented in section 6.2. The proposed mode preselection framework is introduced in section 6.3. Experimental results are given in section 6.4. Section 6.5 concludes the contribution of this approach.

6.2 LITERATURE REVIEW

As discussed in the previous chapters, the computational complexity of an HEVC intra encoder has drastically increased due to the enhanced coding tools, such as 35 prediction modes, larger CTU sizes, quad-tree coding structures, various CU/PU/TU combinations, and time-consuming CABAC rate estimation. All of these enhancements lead to a complex rate-distortion optimization process for HEVC. Inspired by the enormous potential of discovering distinguished features of learning-based approaches, many research efforts have been made to apply machine learning to video compression [51-56].

The researchers in [57] proposed to utilize Convolutional Neural Network (CNN) classifiers to replace the conventional rate-distortion optimization for mode decision in HEVC intra encoders. They use raw pixel values as inputs to the CNNs to avoid data dependencies between neighboring blocks. Therefore, the mode decision of all blocks can be performed in parallel to maximize computational throughput. The proposed neural network consists of two convolutional layers, three ReLUs, one max pooling, and two fully connected layers. This approach has been validated by the HEVC screen content coding (SCC) reference software. The average BD-Rate loss is negligible, only 0.52%. However, the high efficiency of the proposed method in screen context coding does not guarantee similar performance in the general content video coding.

In [58], an intra prediction convolutional neural network (IPCNN) is proposed that relies on the rich context between the current block and its neighboring blocks. Unlike the traditional intra prediction that employs the nearest reference lines to predict the current block, IPCNN adopts three nearest reconstructed blocks to fully utilize the spatial correlation between the current block and its neighboring blocks. The input to the IPCNN contains pixels of three

neighboring reconstructed blocks, while the output is the best prediction of the current block. IPCNN consists of 10 convolutional layers, 9 ReLU layers, and 8 batch normalization layers. Experiments show that the proposed IPCNN can achieve an average reduction of 0.70% in BD-Rate. However, since the proposed network in [58] only supports 8×8 PUs, its efficiency in PUs of other sizes is unknown.

Similar to [58], the researchers in [59] proposed an intra-prediction fully connected network (IPFCN), where all layers are fully connected. This network relies on the spatial correlation between the current block and its neighboring blocks. Particularly, the input is a number of reference lines and the output is the prediction of the current block. Experiments have been conducted to investigate the effects of network depth and size. A three-layer network exhibits the best performance, with 128 nodes per layer. The proposed network has been incorporated into the HM reference software. Experimental results show IPFCN saves an average of 1.1% bitrate. However, this work only investigates intra prediction of 8×8 blocks.

In [60], the authors proposed CNNs to obtain candidate modes for the RDO process, so the original rough mode selection process can be skipped to reduce the computation effort. Moreover, the proposed CNN generates intra prediction modes only by relying on the original pixel values of the current block and the quantization parameter (QP), which means that the CNN mode decision of the current block is independent of the reconstructed neighboring blocks. Therefore, mode prediction of multiple prediction blocks can be performed in parallel to improve the prediction throughput. Experimental results show that this approach can save approximate 28% of the coding time, and the average BD-Rate only increases by 1.15%.

All of the above examples have developed specific learning frameworks to improve prediction performance and reduce the computation complexity of HEVC intra prediction. Like

other research works, we focus on applying machine learning to video compression, regardless of its implementation cost, especially in hardware encoders [61]. So far, little prior studies can be found to develop hardware-affordable neural networks for HEVC intra-frame coding.

In [62], the authors designed a fast CNN-based algorithm that reduces RDO complexity by decreasing the CU partition modes in each CTU. Their algorithm relies on the preprocessing of current-block pixel values, so parallel processing of multiple prediction blocks is realized. The proposed CNN-based intra prediction scheme saves an average of 61.1% of the coding time, while the BD_Rate increases by 2.67%. Moreover, a comparative evaluation of performance and resource consumption has been done in hardware. Experimental results show that the hardware implementation of the proposed CNN using TSMC 65nm technology runs at 714 MHz, whereas the corresponding power and area consumptions are 16.2mW and 42.5k gate, respectively.

Based on the above discussion, we can find that most research efforts have been made to decrease the computational complexity of the RDO process from an algorithm perspective. Considering that after applying CNN-like optimization algorithms, the ultra-high definition (UHD) video coding is too intensive to be implemented in software in real-time applications, there is an urgent need to develop a hardware-oriented neural network framework that can be integrated into hardware video encoders.

6.3 PROPOSED NEURAL NETWORK BASED MODE PREDICTION

In this section, we propose a fully connected layer based neural network (FCLNN) framework [59], which preselects intra prediction candidate modes for each luma prediction block. As the proposed FCLNN is hardware-oriented, it can be integrated into our previously designed HEVC intra encoder to further improve the coding performance in two ways.

First, the FCLNN can improve the intra prediction throughput by removing rough mode

selection (RMS), which has been widely used in intra prediction to reduce candidate modes for the RDO process. In our intra encoder design, we adopt Hadamard transform to select candidate modes for the RDO process based on the residual blocks after intra prediction. Although the Hadamard transform is not as complex as the DCT transform, it still brings some latency to intra prediction due to the large number of prediction modes. Furthermore, due to the data dependencies inherent in intra prediction, the strict correlation of consecutive prediction blocks hinders the high-throughput implementation. To reduce data dependencies, the FCLNN is designed to utilize the original pixels of the current block rather than the reconstructed pixels of its neighboring blocks, so multiple PUs can be processed in parallel regardless of the size and location of PUs. To reduce the latency by mode preselection, the FCLNN is implemented prior to intra prediction in an extra pipeline stage. The throughput of intra prediction can be improved.

Second, the FCLNN is expected to improve coding efficiency in terms of video quality and encoding stream bitrate. In order to improve the coding efficiency of intra encoders, more accurate intra prediction is strongly required. In our intra encoder, Hadamard cost is used to select the candidate modes for RDO process. The Hadamard transform does not guarantee the selection of the global optimal mode, so the RDO result is likely to be a local optimal mode. It is reported that deep learning approaches possess a superior capability in discovering high-dimensional or complex features compared to traditional mathematical patterns. Therefore, it is reasonable to use neural networks for mode preselection.

6.3.1 Proposed Mode Preselection Scheme

HEVC defines 35 intra prediction modes, including DC, Planar, and 33 angular modes. The DC and Planar work well in the smooth regions, while the 33 angular modes are evenly distributed on upper-left reference lines. Angular modes usually exhibit similar predictions over

a small range of angular variations. Inspired by this observation, we propose a candidate mode selection method that includes a smoothness checker, angle detection, and neural networks to preselect intra prediction modes for luma blocks. First, it will always check if the prediction block is a smooth block. A threshold-based sample filter is designed to eliminate homogenous prediction blocks, which trap the neural networks in ill-conditions. Then, an angle detector will be applied to the prediction blocks after the smoothness test. All 33 angular modes will be divided into 7 groups, each of which contains a certain number of highly correlated angular modes. A variable (*Group_Idx*) is produced by the angle detector. This *Group_Idx* is used to select the corresponding FCLNN design, which determines the output prediction mode of the current block. Each FCLNN is dedicatedly trained for a specific mode group.

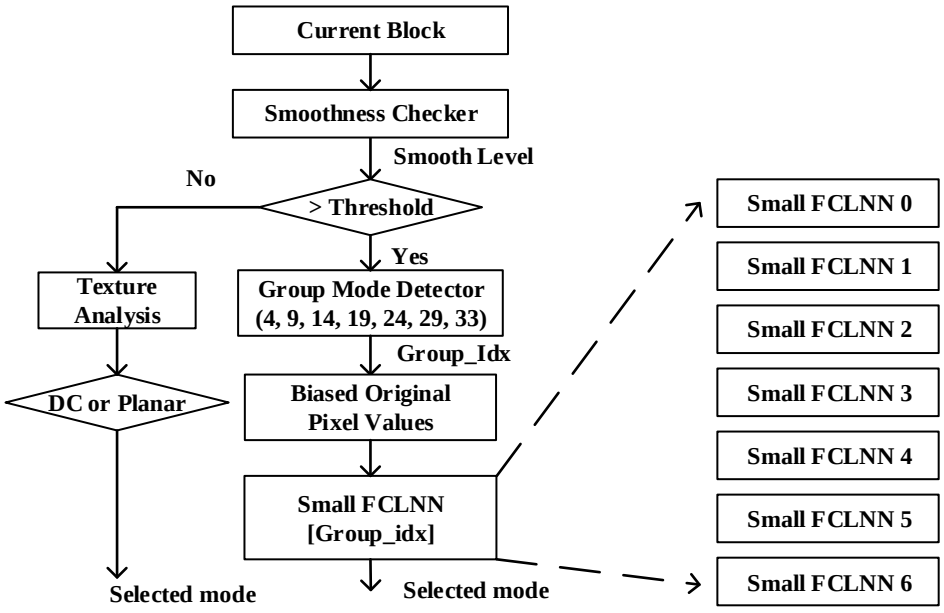


Fig. 6.2. Proposed deep learning-based mode preselection scheme.

Fig. 6.2 shows the proposed mode preselection scheme, which consists of three major components: Smoothness Checker (SC), Group Mode Detector (GMD), and fully connected layer based neural networks (FCLNNs). The SC is used to calculate the smoothness of the current block. A threshold is predefined based on statistical analysis before starting mode

preselection. If the current block is in a smooth region, the candidate mode will be selected from DC and Planar by texture analysis. Because DC is preferred in smoother regions. If the smooth level of the current block is greater than the threshold, the GMD is applied to the current block to find the best mode group. Once the mode group is determined, a dedicated FCLNN is utilized to refine the selection within the mode group. Each mode group contains 3 to 5 intra prediction modes. The input signals to the FCLNN are the biased pixel values, which are derived by subtracting the minimum pixel value from the original pixel values of the current block.

To determine the best candidate mode for each PB, we adopt a two-stage approach in this research. The first stage is a rough selection of mode groups, while the second stage performs mode refinement and selects an optimal mode from the selected mode group. Mode group is determined by examining the prediction results of seven typical prediction modes with original pixel values rather than the reconstructed reference lines. If the best prediction mode is within a specific group, its neighboring modes in the same group are also likely to have better prediction than the modes in other groups. Once the mode group is determined, the FCLNN is used to find the best prediction mode in this group. The design details will be presented in the next section.

6.3.2 FCLNN Based Mode Refinement

In this section, the FCLNN-based mode refinement is presented. Since all 33 angular modes are split into 7 groups and each mode group requires a dedicated trained FCLNN, the number of FCLNNs introduced is also 7. However, due to the angular mode splitting, each FCLNN has less modes to be recognized, compared to the traditional training method. Therefore, the scale of FCLNN can be drastically decreased. Table 6.1 lists the details of angular mode division and the corresponding FCLNN index for each group.

Table 6.1 Details of angular mode splitting.

FCLNN Index	Angular Mode Group
0	2, 3, 4, 5, 6
1	7, 8, 9, 10, 11
2	12, 13, 14, 15, 16
3	17, 18, 19
4	20, 21, 22, 23, 24
5	25, 26, 27, 28, 29
6	30, 31, 32, 33, 34

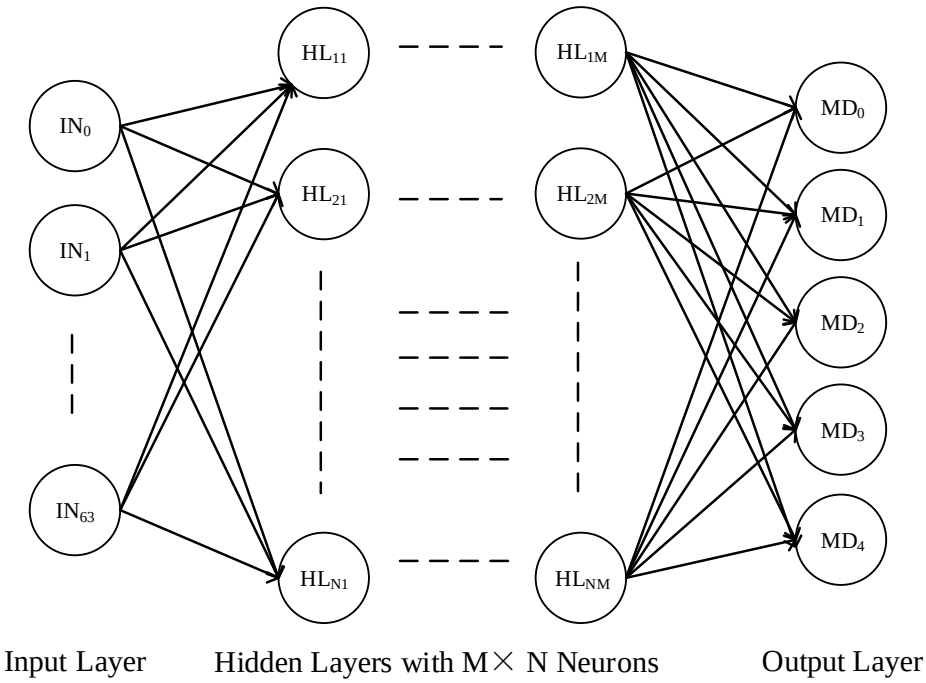


Fig. 6.3. Structure overview of fully connected layer based neural network.

In this dissertation, we propose a fully connected layer based neural network, which is capable of performing mode selection with deep learning capability. As shown in Fig. 6.3, the basic structure of FCLNN consists of one input layer, M hidden layers, and one output layer. The input layer is intended to cope with an 8×8 block that contains 64 biased pixel values. The hidden

layer contains M layers with N neurons per layer. The output layer has 5 output nodes in Fig. 6.3. For mode group 6, the number of output nodes in $FCLNN_6$ is 3. In the following discussion, we will use $FCLNN_0$ as an example.

The basic structure of $FCLNN_0$ is shown in Fig. 6.3. However, the number of neurons (N) and layers (M) remains to be determined. Two experiments are conducted to derive the value of M and N, respectively. In the first experiment, we fix the number of neurons per layer to study the best layer number. Then, we fix the number of layers to investigate the best neuron number.

6.4 EXPERIMENTAL RESULTS

To investigate the best FCLNN structure, we have conducted several experiments. As a preliminary study, we only use 8×8 blocks as our target block for mode preselection in the experiment. Training data and test data are collected from the HM-15.0 reference software. An open-source color image database-UCID [63] has been used as the input video sequence for HM-15.0. If the 8×8 block passes the smoothness checker based on a predefined threshold design, it is collected as a valid training sample. The smoothness checker aims to remove homogeneous blocks of pictures that tend to capture neural network under ill-conditions. Filtered training blocks with corresponding prediction modes will be assigned to different training groups, each dedicated to training a specific FCLNN. Meanwhile, the homogeneous image block will select DC or Planar as its intra prediction mode according to the texture analysis.

Table 6.2 illustrates the training results of proposed fully connected layer based neural network with 2 hidden layers and different neurons. In our experiments, 4 different neuron numbers per hidden layer have been used from 32 to 256. As shown in Table 6.2, with the average increase in neurons, the training accuracy has increased from 59.14% to 88.07%. The experiment was conducted under the following conditions: 200,000 training iterations, 2 hidden

layers, 5,000~20,000 training samples. Obviously, the number of neurons in each layer can certainly improve the training accuracy of the proposed FCLNN.

Table 6.2 Training accuracy of different groups with 2 hidden layers.

Group	N=32	N=64	N=128	N=256
0	50.76%	67.15%	80.60%	91.29%
1	63.08%	63.74%	73.12%	84.07%
2	63.06%	70.90%	81.64%	87.83%
3	71.27%	85.17%	91.66%	92.91%
4	54.85%	76.62%	84.23%	89.07%
5	53.40%	55.85%	68.90%	82.05%
6	57.57%	71.60%	81.78%	89.30%
Average	59.14%	70.15%	80.28%	88.07%

Fig. 6.4 shows the relationship between training accuracy and training effort for all 7 FCLNNs. For each FCLNN, more training iterations will result in higher training accuracy. However, after a certain number of iterations, the improvement in training accuracy becomes trivial. For example, in our experiments, after 300,000 training iterations, the accuracy of each FCLNN becomes stable. The experiments are performed with 128 neurons and 2 hidden layers. It is ineffective to continue training after the neural network converges. The main purpose of this experiment is to investigate the relationship between training accuracy and iteration. More experiments have been conducted to investigate the effects of different hidden layers on training accuracy. For each experiment, we use 300,000 and 500,000 training iterations. Considering the hardware feasibility, the maximum number of hidden layers studied here is four and each hidden layer includes 128 training neurons.

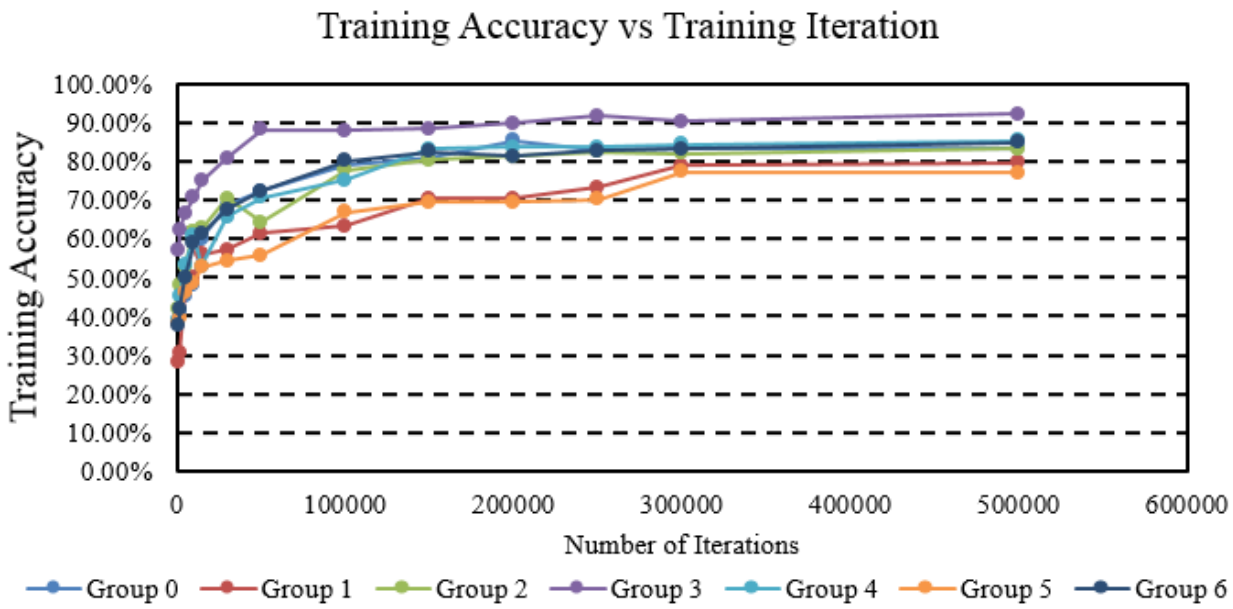


Fig. 6.4. Training accuracy vs training iteration.

The experimental results are illustrated in Fig. 6.5, where more training iterations do not result in more training gains in terms of training accuracy. Regardless of the training layers, increasing the number of training iterations from 300,000 to 500,000 brings negligible accuracy improvements. When the number of hidden layers increases from one to four, the training accuracy has been improved. However, the experimental results with four hidden layers are slightly reduced compared to the training accuracy with three hidden layers. The experiments indicate that a larger network structure does not lead to a better machine learning result. This phenomenon is called overtraining, which is a common issue in the field of machine learning. So far, we have presented several experimental results of proposed FCLNN in mode prediction. FCLNN is the most important part of proposed mode preselction scheme, which is expected to reduce the computational complexity of the RDO process in H.265/HEVC intra encoders. In order to study the best neural network structure from the aspects of layer and neuron numbers, several experiments have been conducted. Hardware implementation should also consider

resource consumption, which is extremely important in low power devices.

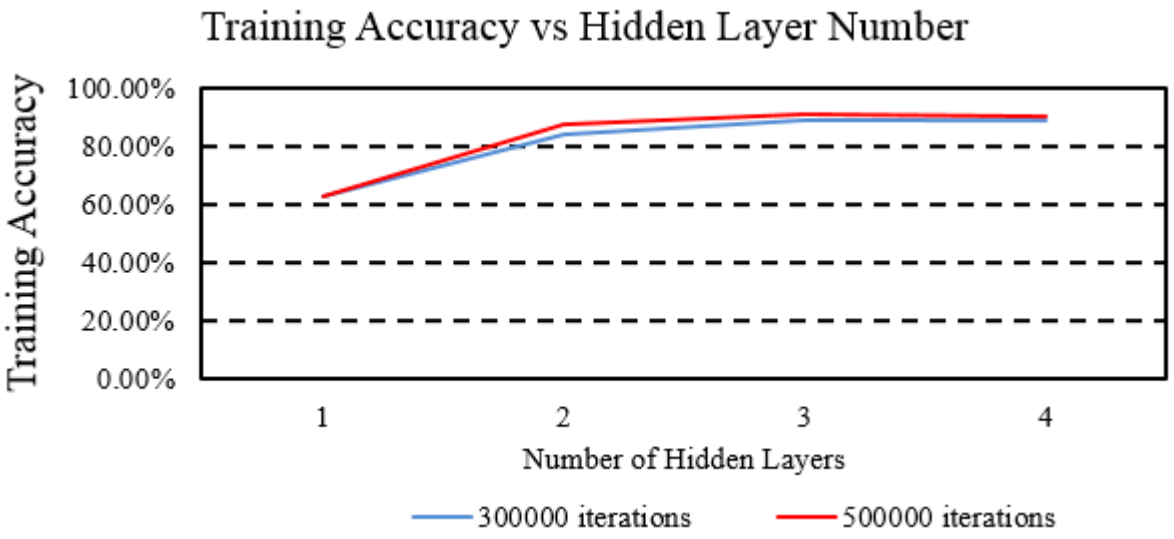


Fig. 6.5. Training accuracy vs hidden layer numbers.

6.5 CONCLUSION

In this chapter, we present a preliminary study on the application of machine learning techniques to video compression. A potential mode preselection scheme is proposed to reduce the computational complexity of HEVC intra encoders. The mode preselection scheme utilizes the mathematical correlation between directional prediction modes and proposes a rough angle detection algorithm. Seven small scale fully connected layer based neural networks (FCLNNs) are employed to refine mode selection after angle detection. Each FCLNN is exclusively trained. 7 FCLNNs have the same neural network structure in order to reduce hardware implementation cost. Experiments show the proposed FCLNN can achieve an approximate prediction accuracy of 80%-90% under different training efforts and neural network structures. The experimental results indicate that machine learning can perform mode identification for video compression. However, when it comes to real hardware implementation of machine learning, different application scenarios will have different concerns in terms of performance, logic area, and power

consumption. For each application, a dedicated neural network structure needs be sophisticatedly designed for optimal performance and cost balance.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In this dissertation, we investigate to efficiently implement HEVC intra encoders in hardware. From the perspective of algorithm, based on the complexity analysis, we propose efficient HEVC algorithm adaptations, which significantly reduce the computational complexity of RDO. The proposed algorithm adaptations exhibit better compression efficiency than existing state-of-the-art designs. Moreover, since all algorithm adaptations have been developed under consideration of hardware implementation, they are friendly to implement in hardware. From the hardware design point of view, we propose a fully-parallel hardware architecture of HEVC intra encoder, which performs the complete cycle of RDO process and entropy coding. In order to parallelize intra prediction of different size PUs, the proposed design consists of four independent prediction engines. To accelerate the rate prediction, a group-based parallel syntax processing scheme is proposed and multiple rate instances are utilized.

The proposed intra encoder demonstrates superior compression efficiency in most scenarios, however, compression efficiency in the worst cases is observed to be relatively low. Further investigation finds that performance degradation is caused by the exclusion of derived luma mode in chroma prediction. Inspired by this observation, we propose to include the derived luma mode. An improved architecture is then developed based on the previous one. To alleviate throughput decreasing due to mode dependency, the clock frequency of rate estimation has been doubled in the new architecture design. The experimental results show our new design exhibits better compression efficiency than the first design with a slight overhead of power and area, meanwhile, it sustains real-time 4K video compression at 30fps.

7.2 FUTURE WORK

The complexity of HEVC video compression is mainly caused by the increased number of prediction modes and partitions. In chapter 6, we proposed to utilize deep learning to predict prediction mode to reduce the computational complexity of intra prediction. This preliminary work shows a learning-based approach is suitable for mode preselection in intra encoder. However, there are still lots of aspects that need to be improved. First, the mode preselection accuracy can be improved. An investigation of the best neural network structure is strongly desired. More experiments have to be conducted. Second, to implement the neural network in hardware, the data format has to be considered in advance. For example, the precision of floating-point numbers can be reduced from 32 to 16, in order to save area and power. Thus, our future work will focus on the study of using machine learning to ease video compression.

REFERENCES

[1] X. Chen, S. Zhang and J. Liu, "Design of UAV video compression system based on H.264 encoding algorithm," *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, Harbin, 2011, pp. 2619-2622.

[2] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, 2012, pp. 3354-3361.

[3] S. Ji, W. Xu, M. Yang and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, Jan. 2013.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.

[5] H. Kawamoto, "The history of liquid-crystal displays," in *Proceedings of the IEEE*, vol. 90, no. 4, pp. 460-500, April 2002.

[6] T. Komine and M. Nakagawa, "Fundamental analysis for visible-light communication system using LED lights," in *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 100-107, Feb. 2004.

[7] S. Kunić and Z. Šego, "OLED technology and displays," *Proceedings ELMAR-2012*, Zadar, 2012, pp. 31-35.

[8] Jun Hyuk Cheon *et al.*, "Active-matrix OLED on bendable metal foil," in *IEEE Transactions on Electron Devices*, vol. 53, no. 5, pp. 1273-1276, May 2006.

[9] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[10] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits Systems for Video Technology*, vol. 22, no. 12, pp. 1668–1683, 2012.

[11] E. Dumic, M. Mustra, S. Grgic and G. Gvozden, "Image quality of 4∶2∶2 and 4∶2∶0 chroma subsampling formats," *2009 International Symposium ELMAR*, Zadar, 2009, pp. 19-24.

[12] B. Ostermann, "Differences between an object-based analysis-synthesis coder and a block-based hybrid coder," *Proceedings. International Conference on Image Processing*, Washington, DC, USA, 1995, pp. 398-401 vol.2.

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

[14] L. Zhao, L. Zhang, S. Ma, and D. Zhao, "Fast mode decision algorithm for intra prediction in HEVC," in *Proc. IEEE VCIP*, Nov. 2011, pp. 1-4.

[15] H. Zhang and Z. Ma, "Fast intra mode decision for High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 660–668, Apr. 2014.

[16] S. Na, W. Lee, and K. Yoo, "Edge-based fast mode decision algorithm for intra prediction in HEVC," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2014, pp. 11–14.

[17] J. Leng, L. Sun, T. Ikenaga, and S. Sakaida, “Content based hierarchical fast coding unit decision algorithm for HEVC,” in *Proc. CMSP*, May 2012, pp. 56-59.

[18] S. Cho and M. Kim, “Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 9, pp. 1555–1564, Sep. 2013.

[19] S. Ma, S. Wang, S. Wang, L. Zhao, Q. Yu and W. Gao, “Low complexity rate distortion optimization for HEVC,” in *Proc. Data Compress. Conf. (DCC)*, pp. 73–82. March, 2013.

[20] J. Zhu, Z. Liu, D. Wang, Q. Han and Y. Song, “Fast prediction mode decision with Hadamard transform based rate-distortion cost estimation for HEVC intra coding,” in *Proc. ICIP*, 2013, pp. 1977–1981.

[21] Y.-K. Tu, J.-F. Yang and M.-T. Sun, “Efficient rate-distortion estimation for H.264/AVC coders,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 5, pp. 600–611, May 2006.

[22] W. Shen, Y. Fan, L. Huang, J. Li and X. Zeng, “A hardware-friendly method for rate-distortion optimization of HEVC intra coding", *International Symposium on VLSI-DAT*, pp. 1-4, Apr. 2014.

[23] S. Johar and M. Alwani, “Method for fast bits estimation in rate distortion for intra coding units in HEVC,” in *Proc. IEEE Consumer Communications and Network Conference*, Jan. 2013, pp. 721–724.

[24] V. Sze, M. Budagavi and G. J. Sullivan, “High efficiency video coding (HEVC): algorithms and architectures,” Springer, 2014.

[25] F. Bossen, “CE1: Table-based bit estimation for CABAC”, *joint Collaborative Team on Video Coding (JCT-VC)*, Document JCTVC-G763, Geneva, Nov. 2011.

[26] Z. Sheng, D. Zhou, H. Sun and S. Goto, "Low-Complexity Rate-Distortion Optimization Algorithms for HEVC Intra Prediction," In *MultiMedia Modeling*; Springer International Publishing: Berlin, Germany, 2014; Lecture Notes in Computer Science; Volume 8325, pp. 541–552.

[27] P. Sharabayko and O. G. Ponomarev, "Fast rate estimation for RDO mode decision in HEVC," *Entropy*, 16 (2014), no. 12, pp. 6667-6685.

[28] J. Zhu, Z. Liu, D. Wang, Q. Han and Y. Song, "HDTV1080p HEVC intra encoder with source texture based CU/PU mode pre-decision," in *Proc. 19th Asia South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2014, pp. 367–372.M.

[29] G. Pastuszak and A. Abramowski, "Algorithm and architecture design of the H.265/HEVC intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 210-222, Jan. 2016.

[30] B. Lee and M. Kim, "A CU-level rate and distortion estimation scheme for RDO of hardware-friendly HEVC encoders using low-complexity integer DCTs," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3787-3800, August, 2016.

[31] X. Zhao, J. Sun, S. Ma and W. Gao, "Novel statistical modeling, analysis and implementation of rate-distortion estimation for H.264/AVC coders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 5, pp. 647–660, May 2010.

[32] S. F. Tsai, C. T. Li, H. H. Chen, P. K. Tsung, K. Y. Chen and L. G. Chen, "A 1062 Mpixels/s 8192x4320p High Efficiency Video Coding (H.265) Encoder Chip", in *Proc. Symposium on VLSI Circuits*, pp. 188-189, June, 2013.

[33] X. Huang, H. Jia, B. Cai, C. Zhu, J. Liu, M. Yang, D. Xie and W. Gao, "Fast algorithms and VLSI architecture design for HEVC intra-mode decision," *Journal of Real-Time Image Processing*, December, vol. 12, no. 2, pp. 285-302, August, 2015.

[34] C. C. Ju, et al., "A 0.5nJ/Pixel 4K H.265/HEVC Codec LSI for Multi-Format Smart phone Applications," in *Proc. International Solid-State Circuit Conference (ISSCC)*, pp. 1-3, 2015.

[35] H. Sun, D. Zhou, L. Hu, S. Kimura, S. Goto, "Fast algorithm and VLSI architecutre of rate distortion optimization in H.265/HEVC," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2375-2390, November 2017.

[36] F. Li, G. Shi, F. Wu, "An efficient VLSI architecture for 4×4 intra prediction in the high efficiency video coding (HEVC) standard," in *Proc. ICIP*, Sep. 2011, pp. 381-384.

[37] D. Palomino, F. Sampaio, L. Agostini, S. Bampi, and A. Susin, "A memory aware and multiplierless VLSI architecture for the complete intra prediction of the HEVC emerging standard," in *Proc. ICIP*, Sep. 2012, pp. 201–204.

[38] C. Liu, W. Shen, T. Ma, Y. Fan, X. Zeng, "A highly pipelined VLSI architecture for all modes and block sizes intra prediction in HEVC encoder," in *Proc. ASIC*, Oct. 2013, pp. 1-4.

[39] A. Abramowski and G. Pastuszak, "A double-path intra prediction architecture for the hardware H.265/HEVC encoder," in *Proc. SDDECS*, Apr. 2014, pp. 27–32.

[40] Z. Liu, D. Wang, H. Zhu, and X. Huang, "41.7BN-pixels/s reconfigurable intra prediction architecture for HEVC 2560×1600 encoder," in *Proc. ICASSP*, May 2013, pp. 2634–2638.

[41] N. Zhou, D. Ding, and L. Yu, "On hardware architecture and processing order of HEVC intra prediction module," in *Proc. PCS*, Dec. 2013, pp. 101–104.

[42] B. Min, Z. Xu, R. Cheung, "A fully pipelined hardware architecture for intra prediction of HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27. no. 2, December, 2017.

[43] H. Azgin, E. Kalali, I. Hamzaoglu, "A computation and energy reduction technique for HEVC intra prediction," *IEEE Trans. Consumer Electronics*, vol. 63, no. 1, pp. 36-43, February 2017.

[44] K. Miyazawa, et al., "Real-time hardware implementation of HEVC video encoder for 1080p HD video", in *Proc. Picture Coding Symp. (PCS)*, Dec. 2013, pp. 225-228.

[45] S. Atapattu, N. Liyanage, N. Menuka, I. Perera, and A. Pasqual, "Real-time all intra HEVC HD encoder on FPGA", in *Proc. IEEE Int. Conf. Application-Specific Syst., Architectures and Processors*, July 2016.

[46] M. Chen, Y. Zhang, C. Lu, "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms," *International Journal of Electronics and Communications*, vol. 73, pp. 1-8, 2017.

[47] Y. Zhang, C. Lu, "Efficient algorithm adaptations and fully-parallel hardware architecture of H.265/HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, early access, 2018

[48] Y. Zhang, C. Lu, "A highly-parallel hardware architecture of table-based CABAC bit rate estimator in HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1544-1558, May 2019.

[49] V. Sze, Y. Chen, T. Yang, and J. Emer. "Efficient processing of deep neural networks: A tutorial and survey." *Proceedings of the IEEE 105*, no. 12 (2017): 2295-2329.

[50] Z. Chen, Y. Li, F. Liu, Z. Liu, X. Pan, W. Sun, Y. Wang, Y. Zhou, H. Zhu, and S. Liu. "CNN-Optimized Image Compression with Uncertainty based Resource Allocation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2559-2562. 2018.

[51] W. Park, and M. Kim. "CNN-based in-loop filtering for coding efficiency improvement." In *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pp. 1-5. IEEE, 2016.

[52] Y. Dai, D. Liu, and F. Wu. "A convolutional neural network approach for post-processing in HEVC intra coding." In *International Conference on Multimedia Modeling*, pp. 28-39. Springer, Cham, 2017.

[53] J. He, W. Yang, and J. Wang. "Fast HEVC coding unit decision based on BP-Neural Network." *International Journal of Grid Distribution Computing* 8.4 (2015): 289-300.

[54] R. Song, D. Liu, H. Li, and F. Wu. "Neural network-based arithmetic coding of intra prediction modes in HEVC." In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-4. IEEE, 2017.

[55] T. Li, M. Xu, and X. Deng. "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC." In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1255-1260. IEEE, 2017.

[56] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan. "Reducing complexity of HEVC: A deep learning approach." *IEEE Transactions on Image Processing* 27, no. 10 (2018): 5044-5059.

[57] T. Laude, J. Ostermann, "Deep learning-based intra prediction mode decision for HEVC." *2016 Picture Coding Symposium (PCS)*. IEEE, 2016.

[58] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, and D. Zhao. "Convolutional neural networks based intra prediction for HEVC." arXiv preprint arXiv:1808.05734 (2018).

[59] J. Li, B. Li, J. Xu, and R. Xiong. "Intra prediction using fully connected network for video coding." In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 1-5. IEEE, 2017.

[60] N. Song, Z. Liu, X. Ji, and D. Wang. "CNN oriented fast PU mode decision for HEVC hardwired intra encoder." In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 239-243. IEEE, 2017.

[61] X. Yu, Z. Liu, J. Liu, Y. Gao, and D. Wang. "VLSI friendly fast CU/PU mode decision for HEVC intra encoding: Leveraging convolution neural network." In *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 1285-1289. IEEE, 2015.

[62] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang. "CU partition mode decision for HEVC hardwired intra encoder using convolution neural network." *IEEE Transactions on Image Processing 25*, no. 11 (2016): 5088-5103.

[63] Gerald Schaefer, Michal Stich, "UCID: an uncompressed color image database," *Proc. SPIE 5307, Storage and Retrieval Methods and Applications for Multimedia*, Dec 2003.

VITA

Graduate School
Southern Illinois University

Yuanzhi Zhang

stephen.zyz@gmail.com

Shandong University, China
Bachelor of Science, Information Science and Technology, June 2011

Shandong University, China
Master of Science, Electrical and Computer Engineering, June 2014

Dissertation Paper Title:
Algorithms and Hardware Co-Design of HEVC Intra Encoders

Major Professor: Dr. Chao Lu

Publications:

[1] Y. Zhang, C. Lu, "High-Performance Algorithm Adaptations and Hardware Architecture for HEVC Intra Encoders", IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, pp. 2138-2145, 2019.

[2] Y. Zhang, C. Lu, "A highly-parallel hardware architecture of table-based CABAC bit rate estimator in HEVC intra encoder," IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, pp. 1544-1558, 2018.

[3] Y. Zhang, C. Lu, "Efficient algorithm adaptations and fully-parallel hardware architecture of H.265/HEVC intra encoder," IEEE Transactions on Circuits and Systems for Video Technology, early access, 2018.

[4] M. Chen, Y. Zhang, C. Lu, "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms," International Journal of Electronics and Communications, vol. 73, pp. 1-8, 2017.

[5] X. Wang, Y. Zhang, C. Lu, Z. Mao, "Power efficient SRAM design with integrated bit line charge pump", International Journal of Electronics and Communications, vol. 70, pp. 1395-1402, 2016.

[6] Q. Huang, Y. Zhang, Z. Ge, C. Lu, "Refining Wi-Fi Based Indoor Localization with Li-Fi Assisted Model Calibration in Smart Buildings", 16th International Conference on Computing in Civil and Building Engineering, 2016.