國立交通大學

電子工程學系　電子研究所碩士班

碩　士　論　文

應用於超高效率 H.265 標準之 高記憶體效率內嵌視訊解碼器

A Memory-Efficient I-Frame Decoder for High Efficiency Video Coding (H.265/HEVC)

學生　：　劉家麟

指導教授　：　李鎮宜 教授

中華民國一〇二年八月

應用於超高效率 H.265 標準之高記憶體效率內嵌視訊解碼器

A Memory-Efficient I-Frame Decoder for High Efficiency Video Coding (H.265/HEVC)

研　究　生：劉家麟	Student：Chia-Lin Liu
指導教授：李鎮宜	Advisor：Chen-Yi Lee

國　立　交　通　大　學

電子工程學系　電子研究所　碩士班

碩　士　論　文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electronics Engineering

August 2013

Hsinchu, Taiwan, Republic of China

中華民國一〇二年八月

應用於超高效率 H.265 標準之高記憶體效率內嵌視訊解碼器

學生：劉家麟　　　　　　　　　指導教授：李鎮宜 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘要

在現今產品裏，電路的趨勢為規模越龐大和複雜，把所有的功能整合在一顆IC當中，便可以使產品輕薄短小甚至是可用於可攜式且具有吸引力的裝置。而當今火紅的應用產品裏頭的視訊壓縮解碼器已從H.264/AVC演進到最新一代High Efficiency Video Coding/H.265，高效率的壓縮演算帶來高效能壓縮卻造成硬體實現上的困難。這些困難，包含了硬體的成本，功率的消耗。在視訊解碼器的記憶體裡，其中的模組包括內嵌預測器(Intra Predictor)、去區塊濾波器(De-blocking Filter)佔了總內部記憶體83%。因此，此作品提出了共享記憶體和記憶體階層兩種方法，實現了低功率低記憶體的內嵌視訊解碼器，改善了記憶體空間，並且具有架構可調整之特色，可支援最新一代影像壓縮編碼HEVC/H.265，使其能夠降低整合的整本，及外部記憶體搬移功率消耗。以視訊解碼器細部來說，我們整合了三個演算模組，內嵌預測器(Intra Predictor)，轉換器(Transform Coder)，去區塊濾波器(De-blocking Filter)搭配記憶體共享，試圖降低記憶體空間，另外更以”預

測”的方法利用記憶體階層來實現改善記憶體頻寬，更可以減少記憶體空間需求，提高預測率，可以高達60%的記憶體節省效率，且可減少記憶體核心功率至19%，以達到低成本低功率的需求。除此之外，內嵌視訊解碼器利用的波前平行處理，達到最高吞吐量8Kx4K@30fps。最後此硬體在以Socle-tech Cheetah Design Kit (CDK)搭配外部記憶體(SDRAM)和CPU的協同合作，能夠在LCD螢幕上播放出正確結果。

A Memory-Efficient I-Frame Decoder for High Efficiency Video Coding (H.265/HEVC)

Student : Chia-Lin Liu　　　　Advisor : Dr. Chen-Yi Lee

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

ABSTRACT

In today’s product, the trend of the circuit is becoming large and complex, to integrate the functionality in an integrated circuit will make the product more portable and more attractive. Nowadays, the video decoder has been revolved from H.264/AVC to High Efficiency Video Coding (HEVC), high efficient compression algorithm brought the better performance but has poor impact on the hardware implementation in the video codec. These conflicts including the hardware cost, power consumption are poor to the chip performance. In the memory requirement, the intra predictor and deblocking filter are occupying almost 83% in the video decoder. As a result, the thesis has proposed line buffer sharing and memory hierarchy methods, and has implemented the low power and low memory requirement I frame decoder. The architecture is reconfigurable, low memory cost and can support the newest video standard HEVC/H.265. Detail speaking, we integrate the intra predictor, transform coder and in-loop filter with proposed memory reduction algorithm, attempting to reduce the memory requirements. We also used memory hierarchy with prediction-based method to enhance the hit rate and lower down the power consumption. Both the line buffer sharing and memory hierarchy methods can achieve the 60% memory reduction ratio with 19% memory core power reduction for the low cost and low power application. Besides, the hardware has been designed using wavefront parallel processing to achieve super-high vision throughput 8Kx4K. Further, the I frame decoder is verified co-design with FPGA platform using Socle-tech Cheetah Design Kit (CDK) with SDRAM and CPU, and the results could be shown on the LCD panel.

誌　謝

在口試結束的那一剎那，我心中的悸動仍無法平靜下來，這一切滿滿的感動都要感謝李鎮宜老師，謝謝老師兩年來總是以親切的微笑，認真的眼神，再加上溫和的言語，一步一步的帶領我走進研究的領域，給予許多寶貴的經驗與建議。我也要謝謝口試委員，陳紹基老師，郭峻因老師，張錫嘉老師，謝謝老師們百忙之中抽空參加學生的口試，給予學生不同思考上的建議與突破。

當然在研究的兩年，最要謝謝 multimedia group 的學長，長宏學長，感謝學長在過去的兩年辛苦的教導，雖然學弟我總是以笨拙的方式來回覆學長的苦心，不過真的都有聽進去。也要感謝李曜學長，總是以慈祥的笑容回答學弟笨拙的問題。這邊也要感謝已經當媽媽的宣婷學姐，雖然學弟曾給 group 的學長姊帶來很多麻煩，也謝謝你們的包容，我會記得你們的教導的。也要謝謝實驗室的其他學長，會帶很多宜蘭餅的志龍學長，常常說我是末代的欣儒學長，很會運動的佳龍學長，籌畫畢旅的人偉學長和思齊，當爸爸的義澤學長，謝謝學長們在研究上給予的鼓勵與建議。當然也要感謝同一屆的好麻吉們，昇展，宇滔，日和常常一起吃飯聊八卦是非，在我失意的時候給予我能力往前進。也要謝謝 SI2 的學長，學弟妹，書餘學長，歐陽學長，秉原學長，紝炘，方如，英秀，鈺筠以及美麗的子菁姊，和伶霞姊，另外還有已經畢業的學長姊，浩民，明諭，智翔，如宏，恕平，博堯，佩妤，淑文。也要謝謝碩二的室友們，男鑫，夏銘，執中，在研究忙完的一天，最開心回到寢室跟室友們聊天，給予我放鬆壓力的好夥伴。也謝謝高中的好同學們，一起在遠從台北到新竹念書一起扶持的奕均，士惟，建彰，家仁，尚謙，宥鑫。

最後，我要感謝我的家人，謝謝我的爸爸和媽媽和哥哥，謝謝你們帶給我無後顧之憂的學習，謝謝你們給我豐衣足食的生活，謝謝你們養我這麼大，你們的恩情就如同大海這麼深，難以報答。也謝謝我的女朋友，于茹，謝謝你在我失意的時候給我安慰，給予我勇氣，給予我往前衝的信念。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 HEVC Standard Overview

In the multimedia world, the technique of the compression is the core of the visual entertainment. Overview of the compression standard established by the well-known institute ITU-T and ISO/IEC is shown in Table 1.1. The ITU-T has built H.261, and H.263 standards used for video telephony. Also, the ISO/IEC organizations had built the MPEG-1 and MPEG-4 visual. Moreover, the two teams jointly cooperate to establish H.262/MPEG-2 which is used in digital storage and H.264/MPEG-4 Advanced Video Coding (AVC). H.264/AVC has been developed as well-known standard all over the world. The evolution of the incoming compression standard focuses on the bit-rate savings at the same video quality. The video consumer products spread in the market broadly such as digital TV, or set-top box, multimedia storage or blue-ray disk. Due to the upcoming consumer electronics like the super-high resolution digital TV or high performance internet video transmission, it is important to develop a new standard more efficient than previous standard. Therefore, on June, 2012, the International Telecommunication Union (ITU) and Motion Picture Expert Group (MPEG) have proposed the newest standard High Efficiency Video Coding (HEVC). Moreover, the primary goal of the newest standard is to achieve 50 percent bit-rate reduction better than H.264/AVC. Its application is suitable for the super-high resolution (8192x4320) digital TV. With the advanced algorithm and high performance coding tools, we believe HEVC will bring the burst contribution to the multimedia world in the future.

Table 1.1: Video Coding Standard including Market and Feature

Standard	Target Market	Target Feature
H.261	Video Telephony	Low Delay
MPEG-1	Digital Storage Media (VCD)	I-B-P Structure Reverse Play
MPEG-2	Video Broadcasting	Interlaced Support
MPEG-4	Video on PC	Objects
H.264	Digital Home Application	Compression Efficiency $>$ 2X
H.265	Ultra-High Resolution	Bit Reduction Rate $>$ H.264 50 percent

1.2 Profiles and Levels

Profiles and levels define restrictions on bit-streams with encoder side and also restrict on the bit-streams with decoder side. In the HEVC standard, three profiles are defined as Main, Main 10, Main Still Picture, each supports particular coding tools respectively and also specifies what is needed in the encoder and the decoder. In the Main Still Picture profile, only intra frames are decoded without using any inter frames. Moreover, in the Main 10 profile, the resolution bit is extended to 10 bits to achieve higher coding performance but with more complexities in hardware implementation. Performance limitations for the encoder/decoder are defined by the set of levels. Therefore, in the Table1.2, the luma pixel rate is 0.5Mpixel/s at level 1.0 to 4Gpixel/s at level 6.2, also the max luma size and bit rate are also listed.

In the HEVC, two configurations are also defined flexibly for the users to choose. One is *High Efficiency*(HE), the other is *Low Complexity*(LC). The purpose of HE configuration is to achieve the best performance in the bit-rate reduction at the same video quality. Therefore, the features of the HE have the most novel coding tools compared to the LC. Contrarily, the LC is targeted for the low-cost and low hardware implementation complexities. Therefore, it lacks of some coding tools for achieving high performance at the same bit-rate savings. In the Table 1.3, the coding tools for the encoder and the decoder between the H.264/AVC, High Efficiency and Low Complexity are listed. The partition size enlarges from 16x16 to 64x64 with flexible coding unit size. Moreover, the Non Square Quad Transform (NSQT) is adopted

Table 1.2: Decoder Level in HEVC

level	Max luma pixel rate (sample/sec)	Max luma pixel samples	Max bit rate
1	552960	36864	128
2	3686400	122880	1000
3	13762560	458752	5000
3.1	33177600	983040	9000
4	62668800	2088960	15000
4.1	62668800	2088960	30000
4.2	133693440	2228224	30000
4.3	133693440	2228224	50000
5	267386880	8912896	50000
5.1	267386880	8912896	100000
5.2	534773760	8912896	150000
6	1002700800	33423360	300000
6.1	2005401600	33423360	500000
6.2	4010803200	33423360	8000000

for the transform size to adapt to the variable inter/intra block size. In the in-loop filter, sample adaptive offset and adaptive loop filter which are the new-born filter machines also exist in the High Efficiency configuration.

Table 1.3: Configurations comparison

Tools	High Efficiency	Low Complexity	H.264
Partition Size	4x4-64x64	4x4-64x64	16x16
PU Partition	Symmetric-Asymmetric	Symmetric	N-A
TU Partition	3-level RQT-NSQT	3-level RQT	N-A
MV Prediction	AMVP MRG	AMVP MRG	Spatial Median
Intra Prediction	Unified Directional Intra (35 modes)LM Chroma	Unified Directional Intra (35 modes)	9 modes
Transform	DCT 4x4-32x32 DST 4x4	DCT 4x4-32x32 DST 4x4	DCT 4x4 8x8
Interpolation Filter	DCT-IF	DCT-IF	N-A
Motion Precision	1/4 pixel 7/8-tap 1/8 pixel 4-tap chroma	1/4 pixel 7/8-tap 1/8 pixel 4-tap chroma	1/2 pixel 6-tap 1/4 pixel bi-linear
In-loop Filter	De-blocking Sample Adaptive Offset Adaptive Loop Filter	De-blocking Sample Adaptive Offset	De-blocking
Entropy Coding	CABAC	CABAC	CABAC,CAVLC

Three types of situations have also been established to be used for different applications in HEVC. *All-Intra* (AI), *Random Access* (RA), and *Low Delay* (LD) are defined for different

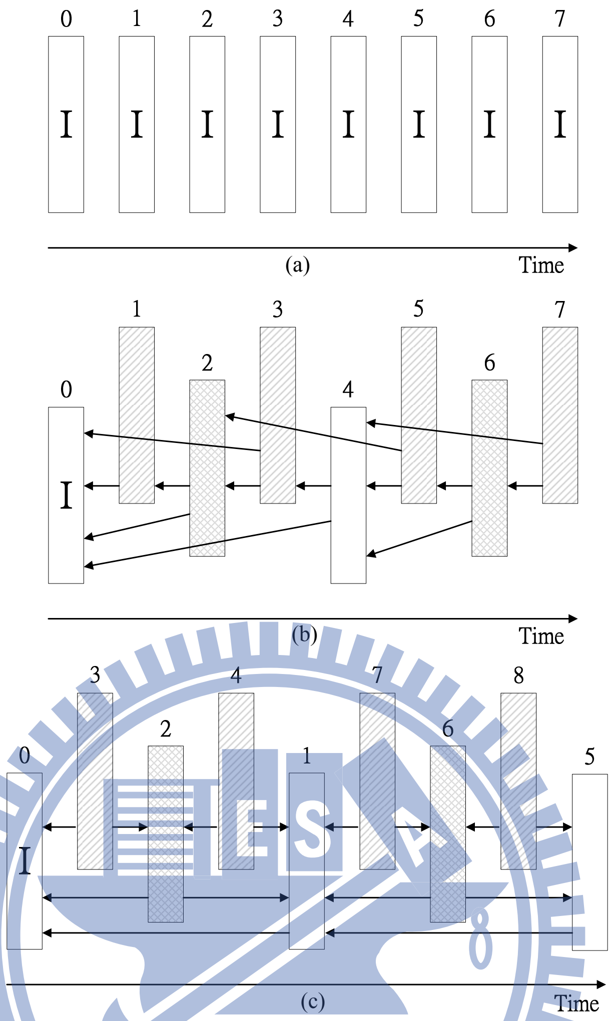


Figure 1.1: (a). All Intra (b). Low Delay (c). Random Access

usage. All-Intra is used for all frames which are intra coded without using temporal domain frames as shown in Figure 1.1 (a). The number on the frame of top is encoding order. Low Delay is for the real time communication with least encoding delay, only the first frame is intra coded, the others are P or B frames using inter prediction as shown in Figure 1.1 (b). Random Access is used for hierarchical structure and I frame is put in the group of pictures periodically, and the encoding order follows the I frame first and then the P/B frames as shown in Figure1.1(c).

The bit-rate saving performance compared with H.264/AVC between each configuration is shown in Figure 1.2 . In this Figure 1.2, we can see that All Intra has little coding gain at almost 28 percent, the most powerful case is random access in the high efficiency condition which can achieve almost 50 percent reduction in bit-rate savings. All the configurations can achieve better bit-rate performance than H.264/AVC.

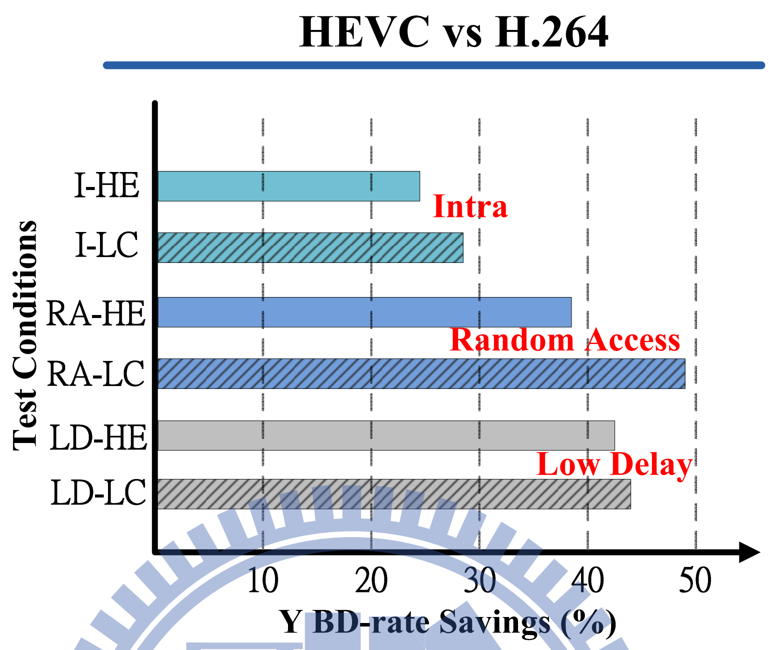


Figure 1.2: Performance in AI, LD, and RA Configurations

1.3 Encoder and Decoder Block Diagram

The encoding process is shown in Figure 1.3. As the input video enters the video encoder, the output will be the binary bit-streams. The encoder is composed of three main parts, one is prediction stage, another is texture coding, and the other is binarization coding. In the prediction stage, intra prediction and motion estimation which are the main contributions in the compression standard are to predict the frames based on the spatial locality and the temporal domain including back and forth. After the predicted frames are produced by one of the predict engines, the output of the minus operation on the original frames and the predicted frames are so called residual data. Residual data then enters into transform coding from the un-apparently the spatial domain to the frequency domain. Due to human eyes insensitivity to high frequency, the data can be quantized and discarded to reduce the reluctant data. Finally, the remaining data will enter into the binarization, entropy encoding process. The purpose of the entropy encoder is to encode the symbols from other modules to the bit-streams. To point why the encoder has the embedded decoder is that due to avoiding the mismatch between encoder/decoder side, the inverse quantization and inverse transform are to reconstruct the residual data. The predicted frames would add with residual data and become reconstructed frames. Due to the motion compensation, reconstructed frames need to match in order to synchronize with encoder and decoder side. Because of the block-based coding, artificial block effect produced by the intra/inter prediction and discrete transform coding, will harm the video quality. Therefore, the filter stage including deblocking filter, sample adaptive offset and adaptive loop filter are to smooth the block edge between boundaries to improve the video quality. The decoder side is

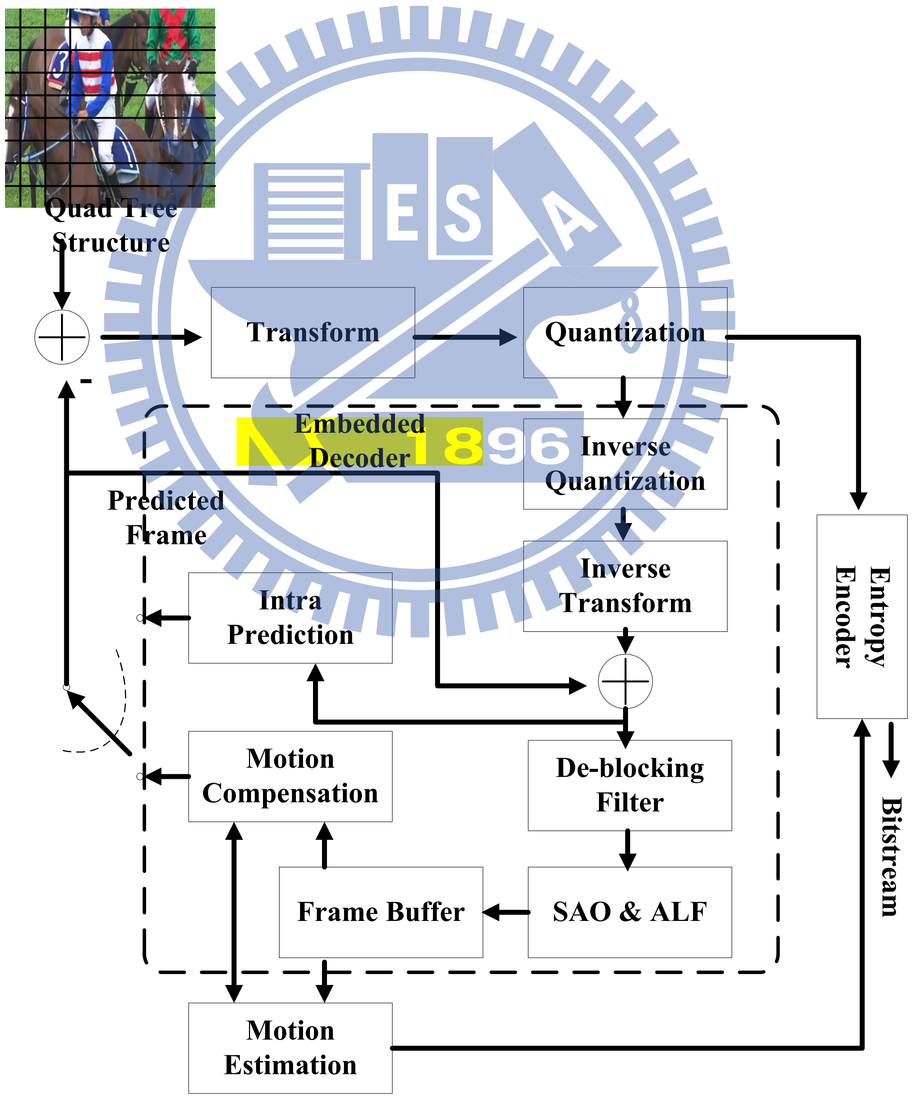


Figure 1.3: Video Encoder Diagram

shown in Figure 1.4 , the main component parts are the same as the encoder. The bit-streams enter into the entropy decoder and produce the syntax symbols and the transform coefficients. The prediction stage and transform coding will be reconstructed by utilizing the synchronous adders. Due to the block edge effect, the in-loop filters are used to improve the video quality in order to smooth the visual badness.

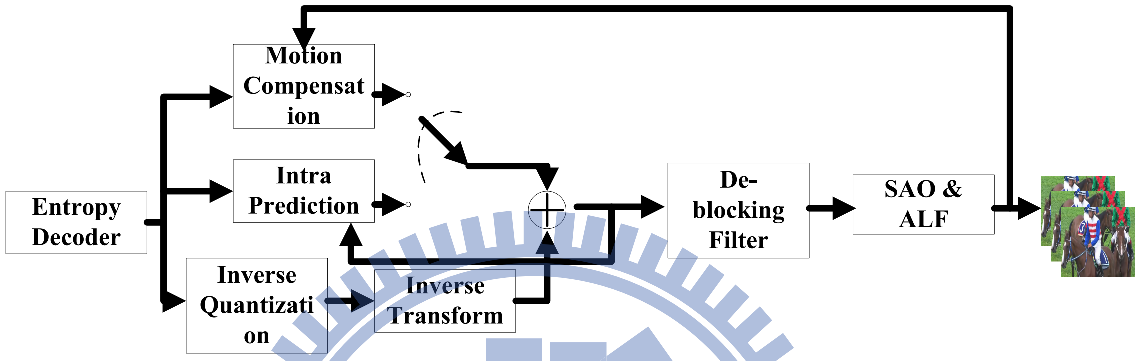


Figure 1.4: Video Decoder Diagram

1.4 Coding Features

HEVC has fruitful novel coding tools different from H.264/AVC described in last section. In entropy coder, the Context Adaptive Binary Arithmetic Coding (CABAC) is still contained in HEVC while the CAVLC is displaced. Due to the high data dependency, the throughput of the entropy coder is limited insufficiently for hardware implementation due to the running frequency. Therefore, the HEVC has adopted a novel technique *wavefront parallel processing* (WPP) which can break out the strong data dependency to improve the system throughput. In the prediction stage, due to the flexible block size partition, the inter prediction has adopted advanced motion vector (MV) prediction not only from the neighboring blocks. Moreover, the DCT-based interpolated filter and motion merge group (MRG) sharing all motion parameters in the adjacent blocks are the main features in inter prediction. In the intra prediction, its angular prediction and adaptive pre-filtering also enhance the prediction accuracy in order to remove more the redundancy. Furthermore, the transform coding structure contains the discrete cosine transform and discrete sine transform to efficiently transform the residuals in to frequency domain. Because of the multiple block-based partition in HEVC, after the reconstruction between transform coding and prediction stage, the artificial defect will be more apparently shown between the block edges. As a result, to further improve quality, the loop filter adds two coding tools sample adaptive offset (SAO) and adaptive loop filter (ALF). The novel filters contain adding offsets and adopting adaptive filter coefficients. The coding tool features will be described in detail in next sections.

1.4.1 Coding Tree Unit Structure

The emerging HEVC standard has adopted flexible block partitions, the coding tree approach in HEVC has brought coding efficiency better than previous standard. In addition, the block size not only supports 16x16 but also enlarges 64x64 with more efficient flexibility. A slice contains multiple coding tree units (CTU) like macroblock in prior standard H.264/AVC. It follows the double z scan order from left to right and top to down in the quad-tree based partitions as shown in Figure 1.5 (a). The CTU has been further split into multiple coding units (CU) recursively. The prediction unit (PU) is contained in the CU, the PU is adopted for inter/intra prediction with more efficient prediction and also can improve the coding accuracy as shown in Figure 1.5 (b). The main part for having better bit-rate reduction is adopted quad-tree PU partitions. Transform unit is for discrete cosine transform, it includes which will decide whether to further split or not.

1.4.2 High Level Parallelism

In the HEVC, the picture is split into many independent tiles which consist of multiple *Largest Coding Units* (LCU). The shape of tiles can be rectangular or square, or the number of tiles can be only on or several LCUs. The coding order of the tile is adopted as raster scan order, also the internal tiles of LCUs are raster scan order too. The concept of tile is like the slice in

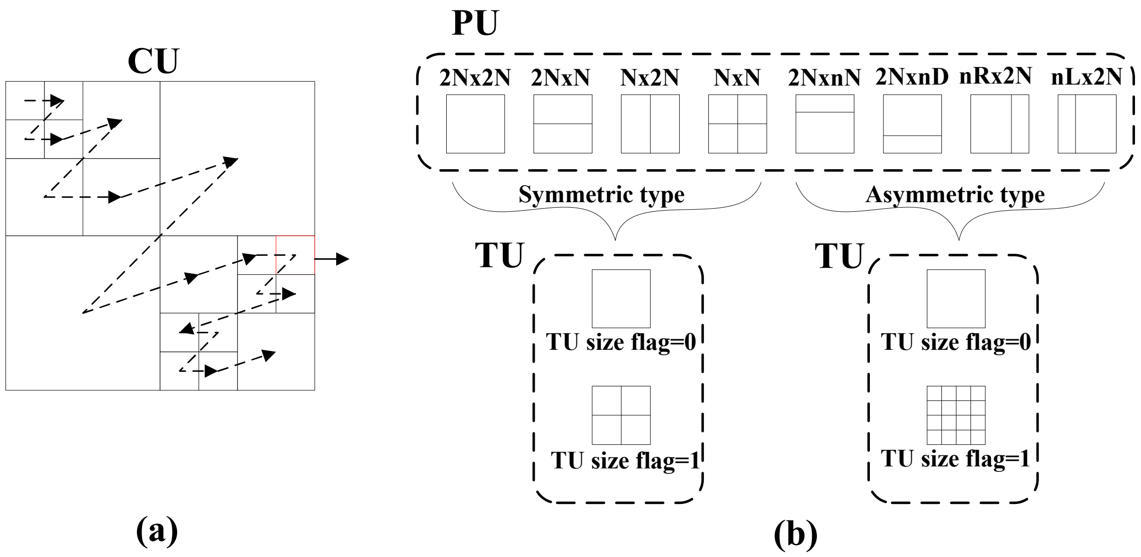


Figure 1.5: CU,PU and TU Partition

the pictures. Each tile could not share the motion parameters such as motion vectors produced by the motion estimation. However, the deblocking filter can filter edge boundaries between the different indepedent tiles for sharing the filter information such as boundary strength. As a result of the broken dependencies between each tiles, the bit-stream encoding/decoding of CABAC can be parallelism to achieve higher throughput without pulling up the frequency in hardware. The coding efficiencies by utilizing parallel CPUs can be apparently gained to adapt to the ultra-high resolution as shown in Figure 1.7. Moreover, the HEVC also has invented a new parallelism technique called wavefront parallel processing. The processing is built at the entropy coding part. As shown in Figure 1.6, in the original entropy part that as the final LCU has finished the coding, the next line of the first LCU can start to encode. Therefore, the strong data dependency tied the throughput of the entropy part. But in the HEVC, the Figure 1.6 points that the entropy coder at the next new line can begin when the top line of 2 LCUs have already been finished. The data dependency would not be followed at the end of the line LCU. Therefore, the multi-thread could enhance the degree of parallelism to improve the throughput and frame rate.

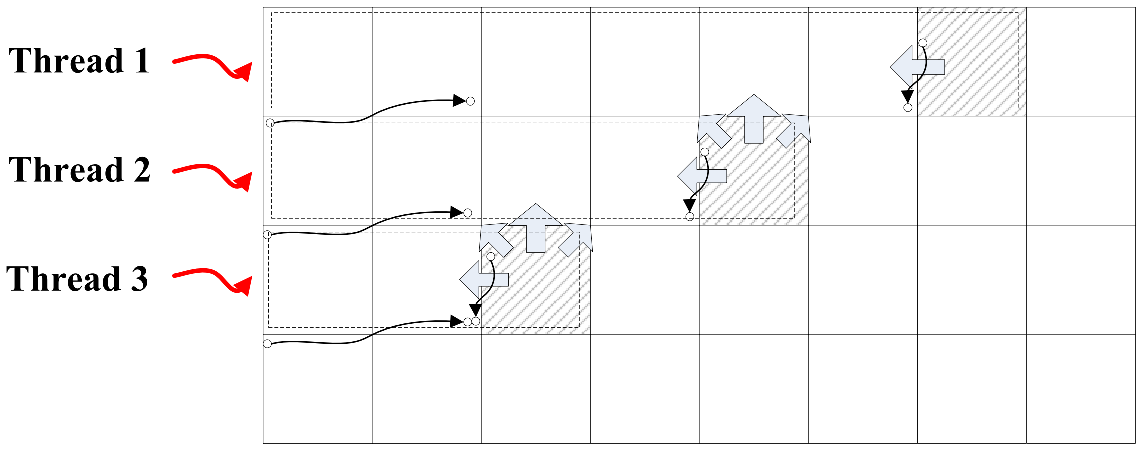


Figure 1.6: Wavefront Parallel Processing in Entropy Coding

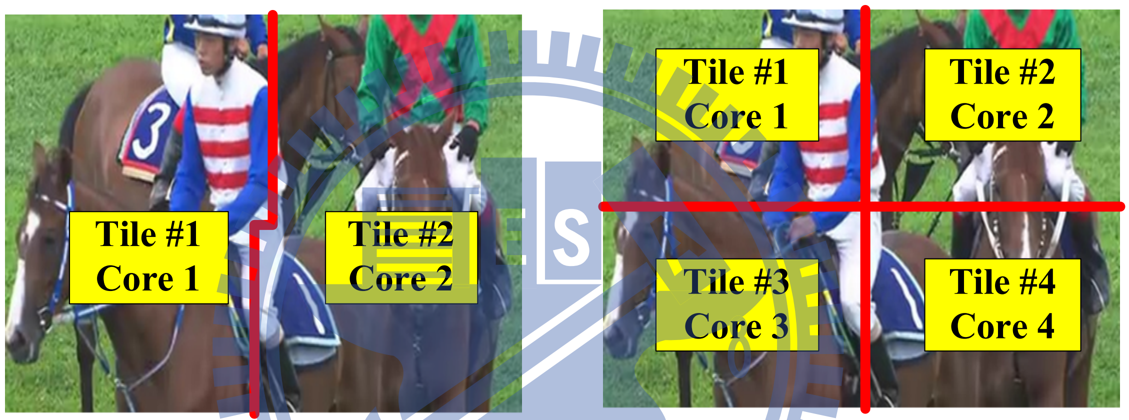


Figure 1.7: Tiles Processing in Multi-Core

1.5 Motivation and Design Challenges

Intra prediction uses spatial correlation to remove the pixel redundancy. At the same time, intra prediction need to utilize the neighboring pixels including upper side and left side. In terms of the video decoder, the neighboring pixels used for intra prediction must be un-filtered before deblocking filter. As a result, with block-based coding in intra prediction, the bottom line of predicted pixels after the reconstruction with IQ/IT needs to be saved due to the lower boundary blocks unavailable. Therefore, the upper line pixels which the line buffer with size of frame width should be stored in the hardware design. Consequently, the intra prediction requires 1 line buffer. Also, deblocking filter's processing order is the same as H.264/AVC, that means vertical edge first and then horizontal edge to gurantee the block boundary is smoothed.

However, in the HM-7.0 reference software design, deblocking filter uses frame-based level coding while it requires large frame buffers storage without concerning the hardware cost and implementation difficulties. This is inefficient in the architecture design, so the macroblock-based coding is likewise adopted in deblocking filter. Therefore, with block-based design, the upper 4 lines filtered pixels which are the neighboring pixels for the deblocking filter should be loaded from the line buffers. After finishing vertical edge filtering, the bottom 4 lines of output filtered pixels would be written back to the 4-line buffers with size still depends on the frame width. Certainly, in the video decoder system, 5 line buffers all depending on the frame width occupy a great portion of the internal memory in the chip area. In spite of achieving lower bit rate compared with H.264 standard with same video quality in HEVC, the coding tools still require the higher hardware cost including memory size, power consumption and chip area. In the mean time, the analysis of the memory requirement for each resolutions shown in Figure 1.8 illustrates that although the storage for deblocking filter dominates the great amount of memory size in the video decoder, intra prediction still occupies almost 15% to 20% percent parts in the higher resolution. Also, in the Ultra-HD resolution, up to 15.36K byte memory is required. Consequently, the equation 1.1 depicts that the whole memory requirements in the video decoder system for the YCbCr (4:2:0). The memory profiling shown in Figure 1.9 is also described that the decoder memory is dominated by the intra predictor and the deblocking filter. Due to the line buffers for intra prediction and deblocking filter are dependent on the frame width, total memory requirement is dominated by intra prediction and deblocking filter. Naturally, how to reduce the line buffers of intra prediction for almost 20 percent memory reduction in the video decoder is an important issue and motivation. Moreover, the memory reduction of the deblocking filter is also important issue in the hardware design.

$$Total = Width(Intra) + 4 \times Width(DF) + Others(Fixed : indepedent) \tag{1.1}$$

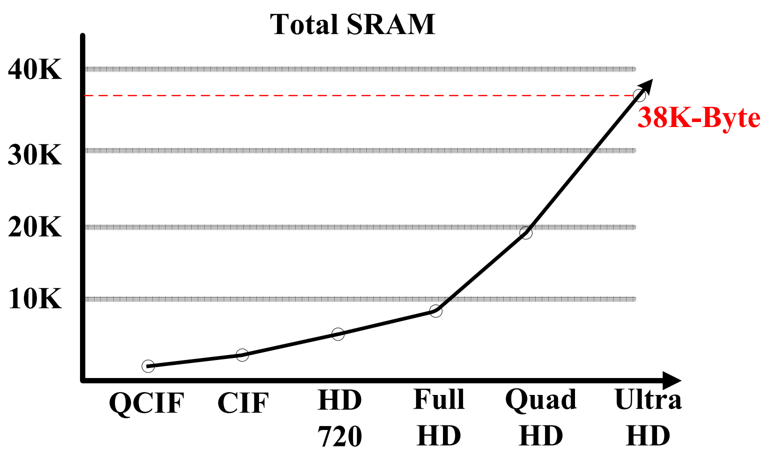


Figure 1.8: Total SRAMs in Video Decoder with Different Resolution

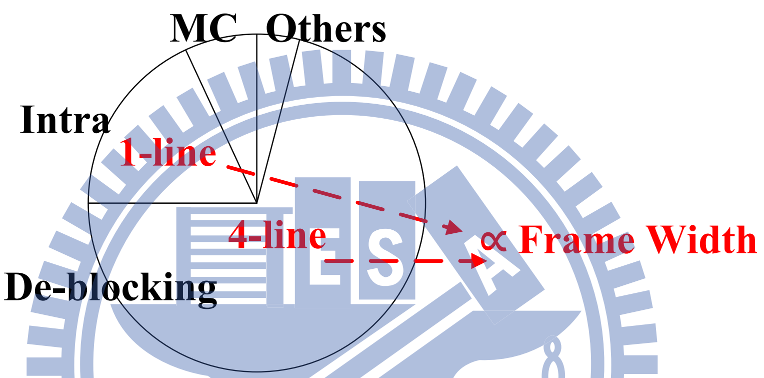


Figure 1.9: Intra and De-blocking occupies 5-line buffer in video decoder

1.6 Thesis Organization

This thesis is organized as follows. In Chapter 2, we study of the I-Frame decoding flow including entropy coding, intra prediction, inverse transform and in-loop filter for the newest standard HEVC and the previous work of the memory reduction will also be stated. Chapter 3 will first introduce the proposed algorithm and compares performance with previous designs. Chapter 4 gives the hardware architecture of the I-Frame decoder system with wavefront parallel processing. Chapter 5 presents implementation results, verification and performance evaluation to verify the hardware design. Finally, conclusions and future works are given in Chapter 6.

Chapter 2

Study of the I Frame Decoding Flow

The study of the I frame decoder will be detail described in this chapter. The I frame decoder of our work shown in Figure 2.1 consists of intra prediction, inverse transform coder, and in-loop filter. The algorithm of each components toward HEVC will be explained later. However, in the low resolution long time ago such as MPEG-1 or MPEG-2, the memory requirement problem is not apparent in the hardware design. However, the applications of HEVC is toward the Quad Full-HD TV, therefore, the memory requirements of the intra prediction and deblocking filter will rise up inefficiently to waste power consumption and area efficiency. In the approaches of [1] and [2], their designs use the external memory and exploit memory hierarchy to decrease the internal memories. The summary would talk about the deficiencies of their designs. The sections below would describe the video component of each functionalities detaily as shown in Figure 2.1.

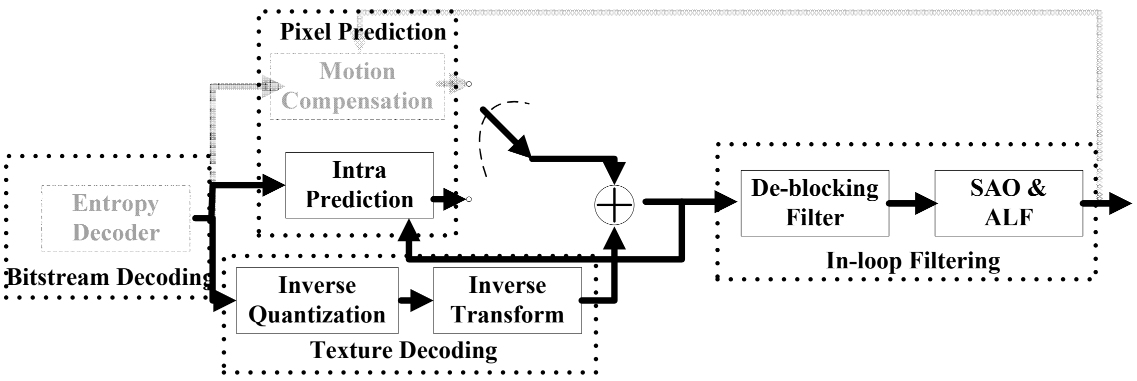


Figure 2.1: Decoder component

2.1 Entropy Decoder

The purpose of the entropy coding is to encode the syntax elements of the video streams in a compressed binary sequence for easily transmittion over the internet or any cable. The elements include the transform coefficients after the DCT and zig-zag scan order, motion vectors which are defined as the difference coordinate of the candidate block and current block, and some picture, slice and block level headers. The elements will be encoded as fixed or variable length binary codes or context-adaptive binary codes based on the element types. When the entropy coding mode equals 0, the transform coefficients will be encoded as context-adaptive variable length coding (CAVLC) by using run-level coding to represent continuous zeros. If the previous elements after zig-zag scan order are numerous 1 or negative 1, the CAVLC will use compact ways to represent 1 or negative 1. Moreover, the non-zero coefficients have strong correlation in the neighboring blocks and the coefficients will be encoded by utilizing look-up table.If the encode coding mode equals 1, the CABAC has better compression performance by adaptively choosing the element probability model. The probability models will adaptively change based on the element contexts. As shown in Figure 2.2, the binary bitstreams will firstly binary arithmetic decode and output the non-binary information bin. Secondly, the context model will update the probability depending on the bins which will be binarized into binary code. If the binary codes are numerous 1s, then the probability will be higher. In the HEVC, the throughput is higher than H.264/AVC due to the following improvements, breaking the model dependencies, grouping the bypass bins and reducing coded bins.

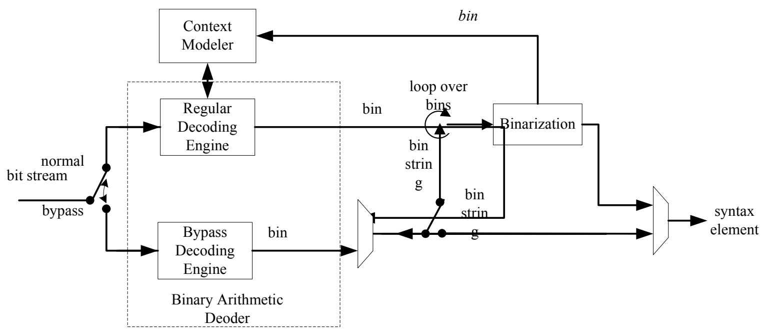


Figure 2.2: Entropy decoder

2.2 Intra Prediction

In the intra prediction, it is one of the important prediction engines in the video coding standard. Since in the H.264/AVC standard, the intra prediction is adopted to improve the spatial redundancy. When the previous frame changed abruptly, like scene change, therefore, the temporal domain prediction will not guarantee the prediction accuracy. Unfortuntely, it may cause the bit-rate higher and would transmit the prediction error. The purpose of the intra prediction is to help the temporal domain prediction to compromise the scene change. Normally, the period of the intra prediction in the group of pictures is the first one every IPPBIPPB or randomly insert an I frame to shut down the error propagation induced by the temporal prediction. With the help of the intra prediction, the H.264/AVC can achieve the hottest video applications during the last decade. In HEVC, the call for proposal during the meeting aims to enhance the bit-rate savings. Therefore, the intra prediction is urgently to further improve the prediction accuracy than previous standard in H.264/AVC. The mode of intra prediction is an important issue to be discussed. In H.264/AVC, the specification defines 9 modes for the 4x4 block, *intra 4x4*, 4 modes for 16x16 block, *intra 16x16* as shown in Figure 2.3. The concept of the intra prediction is to use neighboring pixels to predict the current block. Therefore, the modes in H.264/AVC are sufficiently to predict frames from top, left, to right top corners. However, if the required resolution is Ultra-HD or more, the prediction accuracy for the intra prediction apparently is insufficient. Accordingly, the MPEG team aims to add more modes to further improve the coding efficiency and gain more bit-rate savings. In the specification of HEVC, the modes are defined as 35 more than previous standard as shown in Figure 2.4. Certainly, the results of the predicted frames are more precisely than before. The basic concept of the spatial domain prediction is to utilize the neighboring pixels including top and left reference pixels. The Table 2.1 shows that intra mode number corresponding to the associated names.

In the mode number 1, Intra DC is to average the top row and left column reference pixels same as the H.264/AVC. In addition, the HEVC standard defines the DC post filtering in order to smooth the blocky effect as shown in Figure 2.5(d). In the direct mapping mode such as vertical,

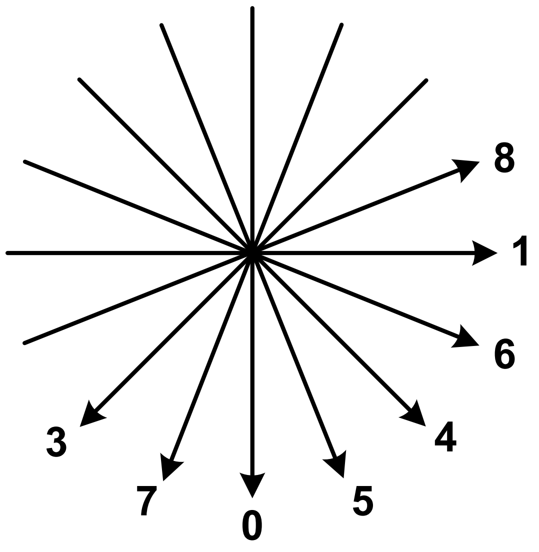


Figure 2.3: Direction of intra prediction in H.264

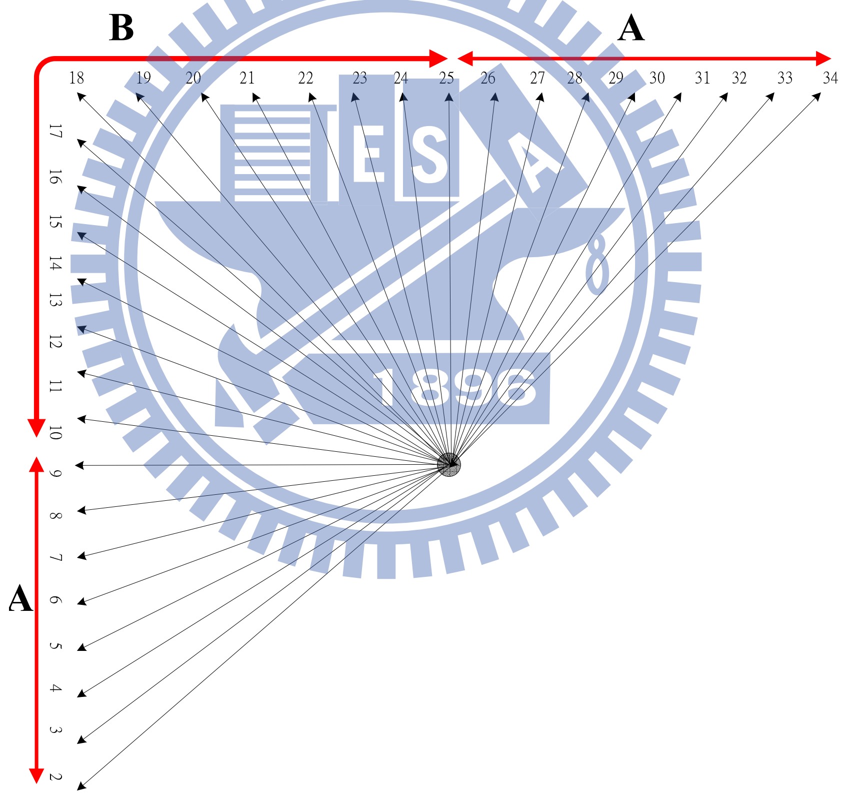
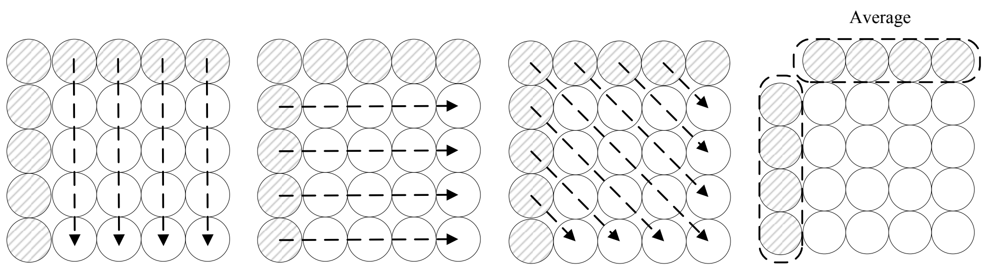


Figure 2.4: Direction of intra prediction in H.265

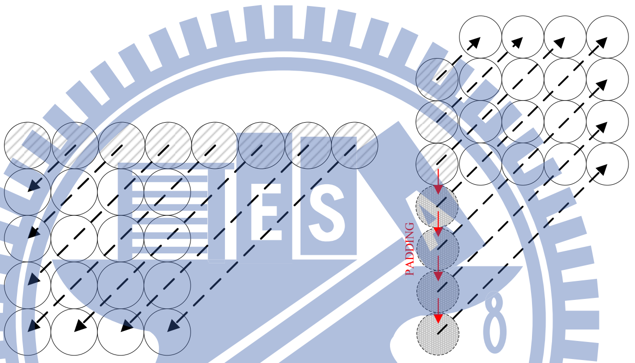
horizontal, diagonal from top right, bottom left and top left are the same as H.264/AVC. In the horizontal-8 mode, if the pixels are unavailable, the pixels are the padding. Also, in order to reduce the blocky effect, after the vertical and horizontal prediction, the post filter between boundary is filtered with predicted pixels and neighboring pixels.In the mode range of 2-34, is

Table 2.1: Intra Mode in HEVC

Intra Prediction Mode	Associate Names
0	*Intra Planar*
1	*Intra DC*
Others	*Intra Angular*
35	*Intra from Luma (chroma)*



(a) Vertical Mode (b) Horizontal Mode (c) Vertical+8 Mode (d) DC Mode



(e) Vertical-8 Mode (f) Horizontal-8 Mode

Figure 2.5: Intra Mode

named as *Angular Directional Intra* (ADI). In the ADI, directions can be classified into two groups. As shown in Figure2.4, the group A uses positive angles while group B uses negative angles. The positive prediction angles have the range from [2, 5, 9, 13, 1, 7, 21, 26], 0 and 32 angles are the direct mapping in the vertical+8, vertical-8 and horizontak-8. The angle is defined as the displacement of the current pixel and top reference pixel in the vertical prediction. Also, it is defined as the right current pixel and left column pixel in the horizontal prediction. We first describe the group A of intra prediction in detail. In Figure 2.6, take mode 27 for example, the current 4x4 block only needs top reference pixels. Different with H.264/AVC, the current predicted pixel needs two inputs to do the linear interpolation. According to the specification, the choice of the two pixels is adopted the angle calculation. As shown in Figure 2.6, the parameter POS_0 will be added with angles which would be achieved by the look-up table to get POS_1, then POS_1 will next to be added with the same angles to get the POS_2. After finishing angle operations, the output of the intra prediction pixel is calculated in 32-tap filtering. If the angles are negative, then the Table2.2 shows the negative angels corresponding to the inverse angles. The processing step to do the intra prediction in negative angles is to first flip the pixels from the side to the main as shown in Figure 2.7(a). Also, the pixel decision is through the look-up table and then do the interpolation as the same as Figure 2.6.

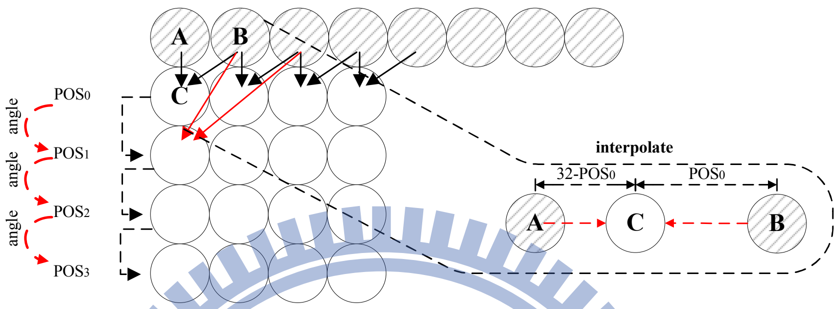
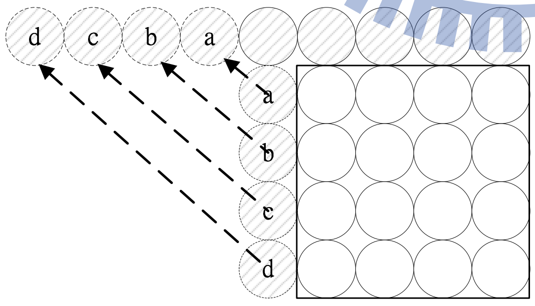


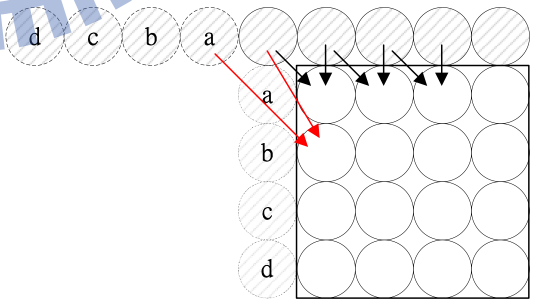
Figure 2.6: 4x4 block interpolation of intra prediction

Table 2.2: Intra Inverse Angles in HEVC

Intra Prediction Mode	2	3	4	5	6	7	8	9
inv Angle	-256	-315	-390	-482	-630	-910	-1638	4096
Intra Prediction Mode	19	20	21	22	23	24	25	26
inv Angle	-315	-390	-482	-630	-910	-1638	4096	



(a) flip the pixels



(b) inverse angle filtering

Figure 2.7: Inverse Angle

To gain the benefit of the homogeneous region in the I frame, HEVC proposed the new planar mode interpolation. As shown in Figure 2.8, the planar operation requires two steps. In the left filtering, pixel a is copied to the rightmost of the current pixel, the leftmost of the pixel is linear operation with pixel a. In the right filtering, the pixel b is also put in the bottommost of the current pixel, the topmost pixel is linear operation with the pixel b. Finally, the bilinear filtering will get the planar output. In concept of the intra prediction algorithm, it requires top row and left column pixels for the input filtering. Further, in the hardware-based design, the coding order follows the raster scan order. Because the next line of LCUs are not available when decoding the first line of LCUs. Accordingly, the intra predicted pixels after the reconstruction with transform data should be stored into the line memory. As the next line of LCUs are ready to decode, the LCU would access the line buffer for the filtering. For not harming the external memory bandwidth, the conventional design requires 1-line internal buffer for storing the temporal pixels. However, the line buffer depends on the frame width would be higher if the H.265 is targeted for the high resolution. How to reduce the line buffer is a big issue in the decoder design discussed in next chapter.

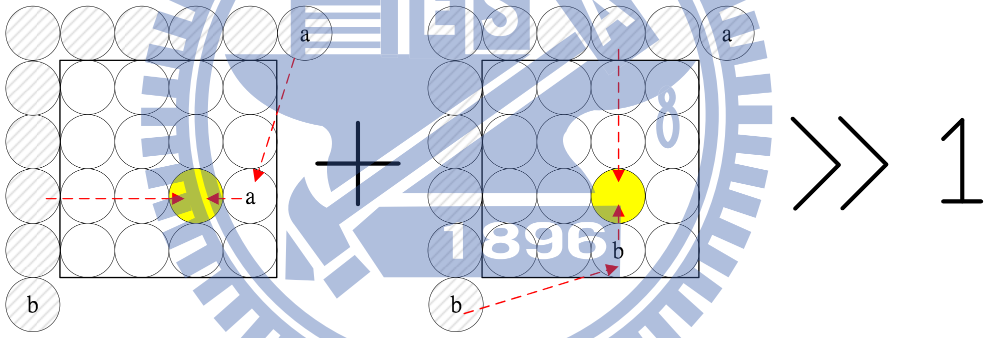


Figure 2.8: Planar Prediction in H.265

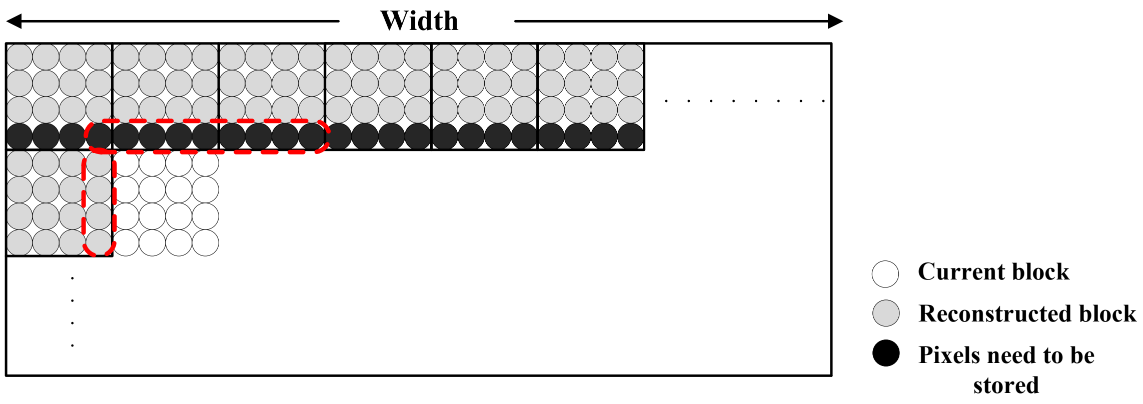


Figure 2.9: Line pixels should be stored into libe buffer

2.3 Transform Coding

Transform coding is widely adopted in video coding to compress the redundant data. As the predicted pixels minus with original pixels in the encoder side, the residual data will be transformed from the spatial domain into frequency domain. Due to the human eyes sensitive in the low frequency, the data in the high frequency can be quantized to discard information so as to compress video data. The transform in H.264/AVC adopts 3 types depending on the intra mode and size. Hadamard transform for 4x4 array of luma DC coefficients in intra macroblock predicted in 16x16 mode. Hadamard transform for 2x2 chroma block and DCT-based transform in 4x4 block. The advantage of the DCT/IDCT in H.264/AVC is that it adopts integer transform which can possibly no mismatch happened in encoder/decoder side. In addition, the core of the transform can be implemented using only shift and additions. While the transform in HEVC has increased up to 32-point transform matrix. Here, the 4-point and 8-point are shown in equation 2.1 and 2.2. The properties of HEVC transform are that the odd rows are even symmetric and even rows are odd symmetric. Moreover, the elements in the smaller transform matrix are also the elements of larger transform. The bold words are also the elements of 4-point transform matrix. The properties make hardware easily implement.

$$H = \begin{pmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{pmatrix} \tag{2.1}$$

$$H = \begin{pmatrix} \mathbf{64} & \mathbf{64} & \mathbf{64} & \mathbf{64} & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ \mathbf{83} & \mathbf{36} & \mathbf{-36} & \mathbf{-83} & -83 & -36 & 36 & 83 \\ 75 & -18 & -86 & -50 & 50 & 86 & 18 & -75 \\ \mathbf{64} & \mathbf{-64} & \mathbf{-64} & \mathbf{64} & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ \mathbf{36} & \mathbf{-83} & \mathbf{83} & \mathbf{-36} & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{pmatrix} \quad (2.2)$$

2.4 In-Loop Filter

Due to the block-based coding structure adopted in the video compression standard, the discontinuities between the block edge cause the blocking effect. The major source of the blocking effect comes from the prediction stage, transform coding and quantization step. Due to the multiple shapes inter blocks adopted during the H.264/AVC and HEVC, the block discontinuities exist followed by the predicted block. Also, the transform coding adopted block-based split structure, the inverse discrete cosine transform cannot recover back the transform coefficient completely in the decoder side. Moreover, in the quantization process, the step of the transform coefficient aims to reduce the high frequency redundancy. However, in the reverse quantization which is also called the scaling process cannot scale back completely. Above the reasons, the MPEG team proposed filters which can smooth the blocking effect between block boundaries. Three types of the filters, one is traditional deblocking filter, another is sample adaptive offset which would add offset to the deblocked pixels, the other is adaptive loop filter which utilize the adaptive filter coefficients in the encoder side, they are adopted in the HEVC which already improved the video quality better than previous standard H.264/AVC.

2.4.1 Deblocking Filter

The differences of H.264/AVC and HEVC in deblocking filter are shown in Table 2.3. In the filter size, H.264/AVC adopted 4x4 filter size to guarantee that every edge would be smoothed. In contrast to the H.264/AVC, being suitable for the high resolution, if the filter edge is still 4x4 size then the complexity and coding time will be sufficiently large than H.264/AVC. Accordingly, the HEVC decided to filter in 8x8 size not only to support larger resolution but also enhances the coding efficiency. Moreover, to reduce the complexity in software/hardware implementations, the boundary strength of the deblocking filter decreases to 0$\sim$2. 0 means to skip the deblocking filter, 1 means normal edge effect, 2 means strongest edge effect. In the filtering order, it is the same as H.264/AVC from vertical edge first and then horizontal edge in the LCU. The coding feature of the HEVC comes from the highly parallelism, wavefront processing and tiles. In the deblocking filter stage, the parallelism is also adopted to improve the coding efficiency. It still keeps the data dependency which is caused in the filtering order, vertical edge first and then horizontal edge. In the software-based implementation, the deblocking filter operates vertical edge filtering in the frame-level and then horizontal edge filtering will start after the frame finished vertical edge filtering. This frame-level parallelism makes deblocking filter more efficient and more fast while the strong data dependency has been broken down. The main differences about the in-loop filter in H.264/AVC is that HEVC creates two coding modules to further improve the video quality, the sample adaptive offset which will first classify the deblocked signals and add the specified offsets to the group signals, the other is adaptive loop filter, it uses wiener filter to decrease the mean sum error between the original data and filtered data, the multiple shapes of the filter are its advantage to flexibly merge into each region. The details of the algorithm will be discussed later.

The blocking effect which is caused by the block-based coding unit is easily noticed by the human eyes. However, not every block edge should be smoothed by the deblocking filter. Redundant smoothing will cause the frame fuzzing, also cause the coding inefficiently. Therefore,

Table 2.3: Deblocking filter in H.264 and H.265

	HEVC	H.264
Filter Size	8x8	4x4
Filter Order	Vertical to Horizontal	Vertical to Horizontal
Boundary Strength	0,1,2	0,1,2,3,4
Parallelization	Yes	No

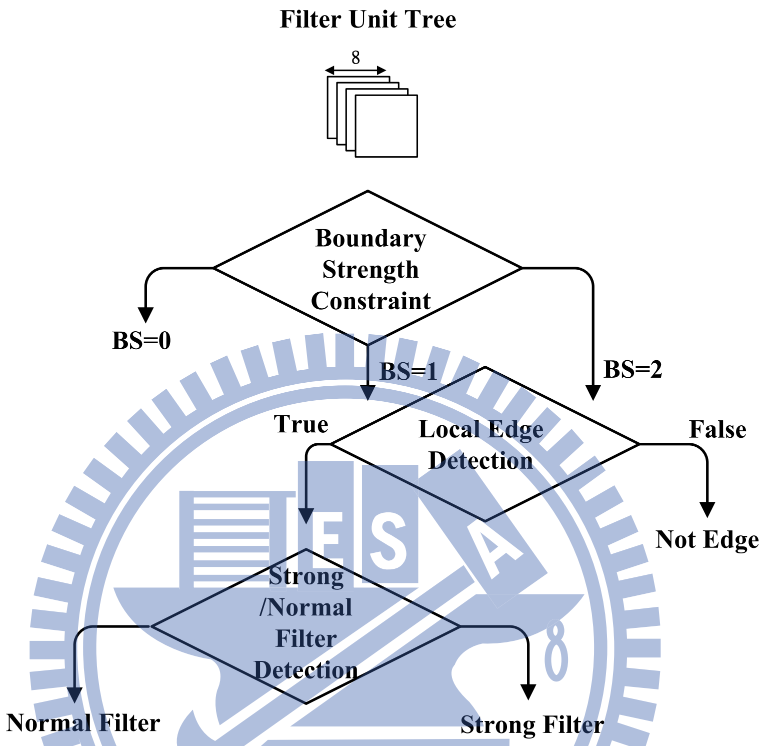


Figure 2.10: De-blocking filter in the filtering flow

the decision stage before filtering is vital to the deblocking filter. The filtering stage is classified for 3 steps, one is boundary strength calculation, another is local edge detection and the other is normal/strong filter decision as shown in Figure 2.10. In the boundary strength classification, the strength number means how strong the blocking effect arise on the block edge. In H.264/AVC, the strength number counts for 0$\sim$4. In HEVC, in order to achieve low complexity, the strength number is reduced to 0$\sim$2. In Table 2.4, the filter condition is that if the samples in the block meet the condition, then it corresponds to the boundary strength. In 2, if the samples in the block are intra coded, then the block effect is the strongest which means the edge is over-sharped. In 1, if the samples of block p or q in a block unit has non-zero transform coefficient and boundary is transform unit or use different reference pictures or different number of

Table 2.4: Deblocking filter boundary strength in H.265

Filter Condition	*Boundary Strength*
The sampls p or q in a coding unit is intra prediction	2
The sampls p or q in a coding unit non-zero tranform coefficients and boundary is transform unit or use different reference pictures or different number of motion vectors	1
Others	0

motion vectors, then the edge is normal effect often seen in the P or B frames. Otherwise, the deblocking filter is skipped for maintaining the edge contents. After the boundary strength is decided, if it is greater than 0, the following local edge detection is performed. In the local edge detection, its purpose is to guarantee whether the edge is exactly existed or not. In the Figure 2.11, the P block and Q block is 4x4 block size, and the equations of (1)∼(3) are the detection process. The $dp0$ is following the sobel operation to achieve the operation results. After the sum

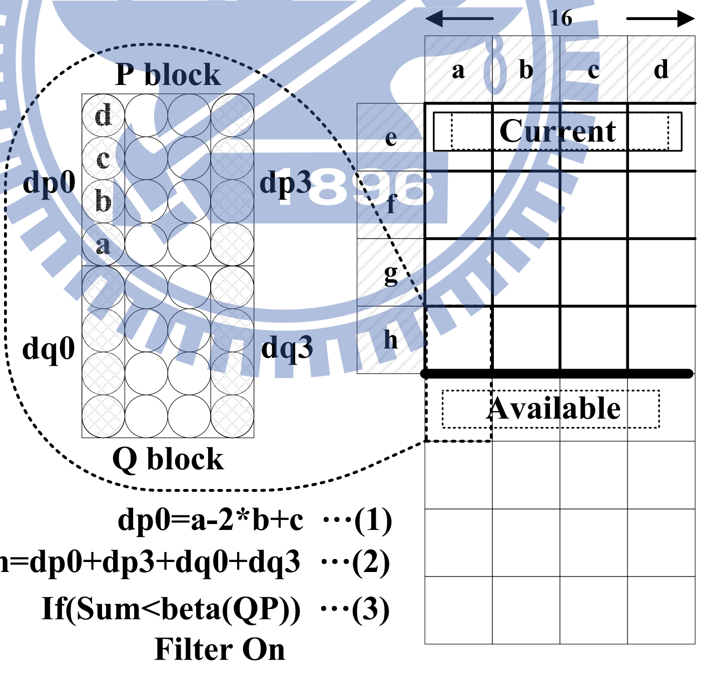


Figure 2.11: Operation in deciding the filter on

of the $dp0$, $dp3$, $dq0$, $dq3$ in the P or Q block, the comparison with the beta and sum will decide whether the edge would be filtered or not. The parameter beta is dependent on the quantization step Q. The listed Q and beta are shown in the HM-7.1 reference software. As the sum is less than beta, then the filter is actually on. Summarily speaking, the deblocking filter is almost the same function of the previous standard with less hardware complexity.

2.4.2 Sample Adaptive Offset

In the sample adaptive offset, the main point is to further reduce the distortion of the deblocked pixels. The concept of the sample adaptive offset is to classify the reconstructed pixels into multiple categories. Each category corresponds to different offset. After the specified classification, the offset finally adds with the deblocked pixels. The offset of each region and classification type will be coded into the bitstreams. The HM-7.1 reference software reported that SAO achieves 3.5 percent bit-rate reduction and up to 23.5 percent bit-rate reduction with less than 1 percent encoding time increase. The followings are going to describe the algorithm of classifications in the SAO. The offset may differ sample by sample in the same region. In the low complexity configuration, two types of classifications are adopted in HEVC. The first is *Edge Offset*, the other is *Band Offset*. In the Band Offset, the main method of classification is to separate the pixels by the gray-level intensity into multiple regions as shown in Figure 2.12. In the 8 bits of gray-level, it ranges from 0 to 255, each sample will be classified into one

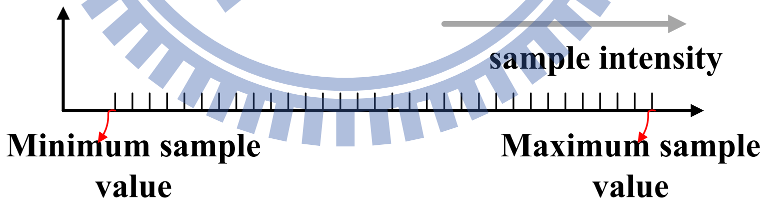


Figure 2.12: Sample Classification in HEVC

region. Therefore, the region of width is 8, totally 32 regions will separate the signals from low intensity to high intensity. In the Figure2.13, the dotted line is original curve, the dash line is the distortion curve which has to be added with offset to pull back the curve. In the bank k and k+2, the positive offset is to pull up the distortion curves to the almost original curves, and in the bank k+1, the negative offset is to pull down the over-intensity signals to the appropriate curves. Besides, the region of index and offset will be signaled to the decoder to lessen the burden of

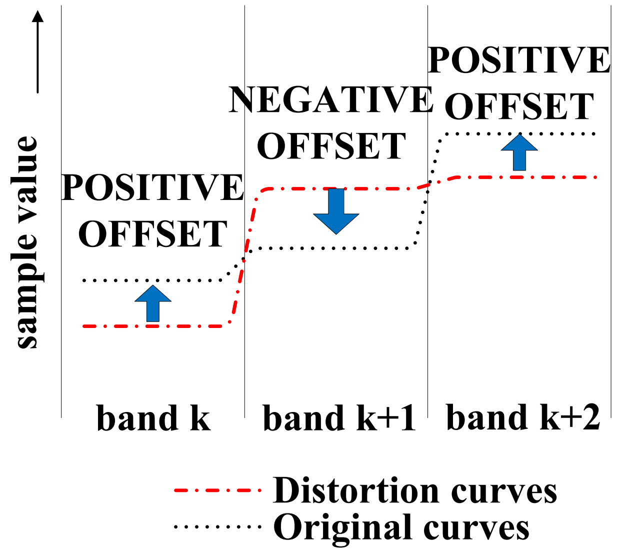


Figure 2.13: Sample Classification in HEVC

the coding complexity. In the Edge Offset, the purpose is to compare with neighboring pixels as shown in Figure 2.14. There are 5 categories which are listed in Table 2.5. In the category 1,

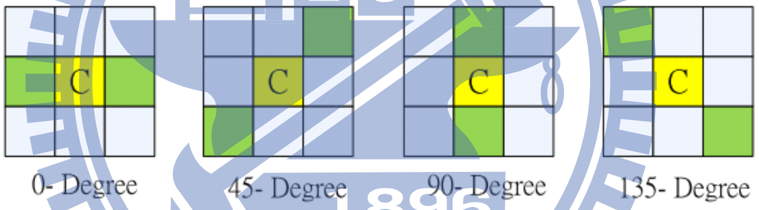


Figure 2.14: SAO Edge Offset Type in 4 mode

Table 2.5: Edge Offset Category

Category	*Condition*
1	c<2 neighbors
2	c<1 neighbor and c==1 neighbor
3	c>1 neighbor and c==1 neighbor
4	c>2 neighbors
0	None of the above

if two neighbors are larger than current pixel c, then positive offset will be added to pull up the intensity level. In the category 2, if the condition is true, then the positive offset is also added. In the category 3 and 4, the negative offset are going to pull down the current pixel to smooth the pixel value. As shown in Figure 2.15, the positive offset in the 1, 2 and 3 are shown to pull up the over-low intensity between the left and right pixels. In addition, the negative offset in the 3, 4 and 5 are to pull down the over-high intensity signals.

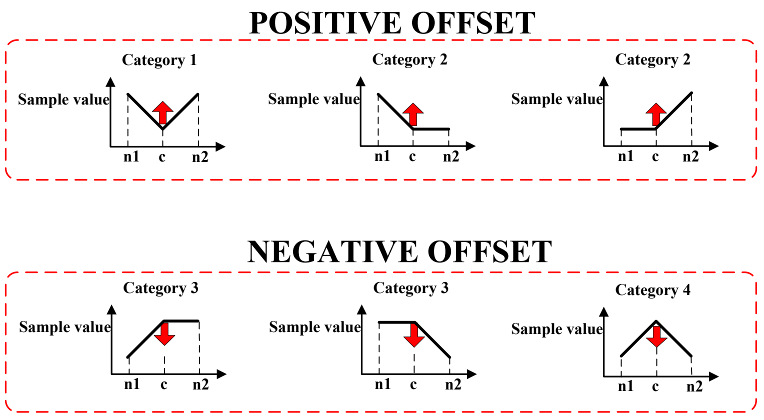


Figure 2.15: SAO Offset type

2.4.3 Adaptive Loop Filter

The Adaptive Loop Filter of the incoming video standard HEVC is adopted in the in-loop filter of the final stage. It is applied after the deblocking filter and sample adaptive offset in order to further reduce distortion compared with original frame. Its innovation is to use *Wiener filter* to reduce *Mean Square Error* related to the reconstructed frames. The algorithm of the Wiener filter filter out the signals which are already polluted. Based on the communication theory, the MMSE estimator is built on the wiener filter coefficient construction. The coefficients generated by the Wiener filter are adaptive to the pixel content region. It means that in the encoder, the loop of the producing filter coefficients is the time consuming due to the pixel content variable. In the encoder, over 40% of the time is on the statistic data gathering. Therefore, the reduced time complexity is a big issue in the implementation. The Figure 2.16 shown below is to describe the differences between the original frame and the reconstructed frame should be the least during the recursively wiener filter coding loop. The filter shape has multiple 2 dimensional filters with cross diagonal. In the encoder side, the filter unit splitting is complete with recursive structure. The Figure2.17 shown below is that the filter on/off is encoded recursively with quad-tree structure. Also, the encoded parameters including filter on/off and filter coefficients

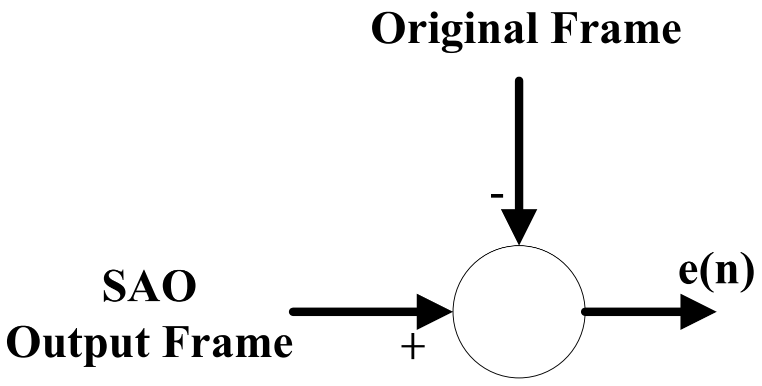


Figure 2.16: ALF Concept

will be signaled to the decoder to lessen the decoder complexity. In the cross filter shape, as shown in Figure 2.18. The number in the block means the same coefficient index which means that as the product of the coefficient and the pixel value, the sum of the products would finally clip to the appropriate value. The ALF in the decoder side is as described follows. First, parse

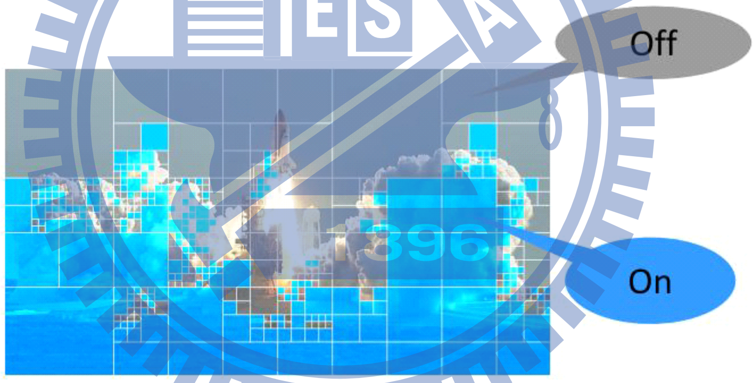


Figure 2.17: ALFOn or Off

the syntax element from the entropy decoder from the bitstream to decode the partition diagram which is filter on/off and also construct the filter coefficients and the filter type. Second, map the filter parameters into pixel regions to reconstruct the filter outputs. Thirdly, store the filtered pixels to the temporary buffers because the filtered pixels would not be the filter input.

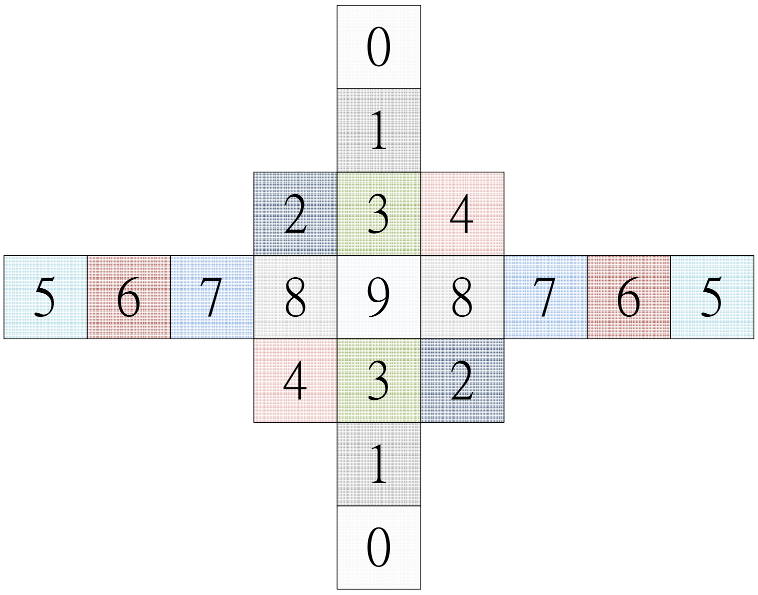


Figure 2.18: ALF filter shape

2.5 Related Works of the Low Memory Architecture

The intra predictor and in-loop filter are the memory dominated coding tools in the video decoder system. There are total 5 line buffers all depending on the frame width which occupies a great portion of the internal memory in the chip area. In spite of high coding efficiency in HEVC, the coding modules still require the higher hardware cost including memory size, power consumption and chip area. In the meantime, the memory requirement for the video decoder hurts the performance of the I frame decoder. In the intra prediction, it originally occupies almost 15$\sim$20 percent parts in the higher resolution. Also, in the Ultra-HD resolution, up to 8 Kbytes memory are required. For the deblocking filter, the macroblock-based coding results the bottom 4 line of pixels not horizontal filtered. Therefore, the 4 line of pixels should be stored for the next new line of LCU is ready. Consequently, in the next section, the traditional approaches are aimed at discussing the intra prediction and deblocking filter memory usage. Due to the line buffers for intra prediction and deblocking filter are dependent on the frame width, total memory requirement is dominated by intra prediction and deblocking filter. Naturally, how to reduce the line buffer of intra prediction for almost 20 percent memory reduction and also the 75% deblocking filter memory in the video decoder are important issue and motivation.

2.5.1 Traditional Approaches

The memory in the design [3] is aimed at HD real-time decoding which the intra predictor uses size of frame width with BRAM cache to store the data. The main contribution of the [4] is to enhance the maximum throughput, which can reach 1991Mpixles/s for 7680x4320p. However, the memory for the intra predictor requires almost 15Kbytes which occupies the core chip area 20 percent. In the traditional approaches of the intra predictor, almost all designs use 1-line buffer to store the data.

For the deblocking filter approach, the design of the [5] has declared two SRAMs, the one is 144x32 bits single-port SRAM, the other is 16x32 bits two-port SRAM, also they exploit group-of-pixel in the memory store instead of column-of-pixel or row-of-pixel. Also the other work of the [6] has required two-port 160x32 bits SRAM to store the current macroblock and adjacent temporary filtered pixels. The design utilizes bus-interleaved [7] to improve 7x performance throughput while they exploit the emulated ARM cpu and embedded SRAM with size 96x32. The work also utilizes three on-chip SRAM modules to store the luma, cb and cr data in the [8]. Also, the architecture uses two-port SRAMs with size 16x32 in [9]. However, the external memory bandwidth problem did not make point to solve. Therefore, make all the pixels out of the chip memory is not a good solution for the deblocking filter. Above designs although did not use 4-line buffers to store the temporary data, the external memory bandwidth is a serious problem happened in the system. External memory bandwidth causes power consumption and also reduce the system throughput. In [10], the design requires frame size data buffer for storing the Line-of-Pixel(LOP). Therefore, the 4-line data buffer is huge compared with above designs which take on-chip SRAMs off the chip, but it does not need any external memory bandwidth. In most designs in intra predictor and deblocking filter, they did not analyze the trade-off with the external memory bandwidth and the internal memory requirement. Some of designs target for the content buffer only using external memory, while others use frame-dependent size memory to neglect the SRAM size wasting. The next section will describe the methods to reduce internal memory with the system viewpoint considering the power consumption and also the memory bandwidth.

2.5.2 Line-Pixel Lookahead

The design improves the memory hierarchy and reduce the embedded SRAMs for the intra prediction, deblocking filter for achieving low power consumption in the [1]. The design aims at copying the correlated data from larger memories which has high data-correlation to the smaller memories. This concept enhances the access time latency and the embedded SRAM could be set smaller with appropriate hit rate. The memory hierarchy adopts three-level including the content, slice and DRAM. The slice memory allocates the all row reconstructed pixels for the intra prediction and vertical filtered pixels for the deblocking filter. Further, they also proposed the line-pixel-lookahead to eliminate the un-used pixels. The main idea of LPL scheme is to

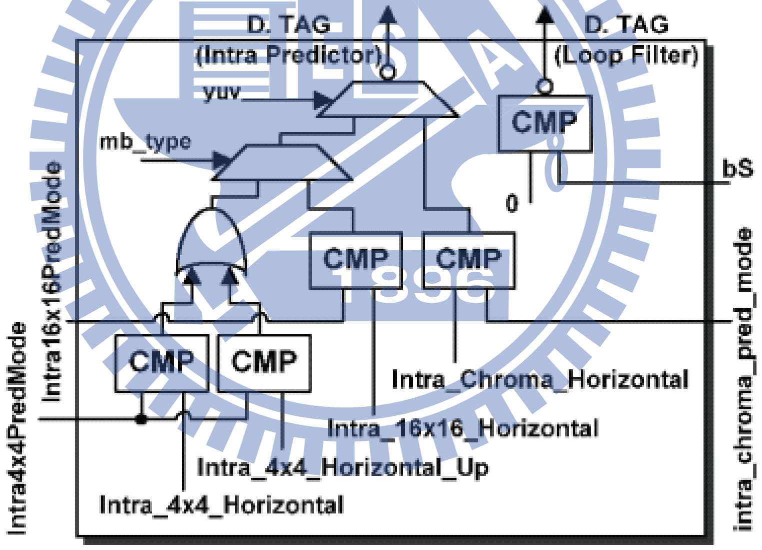


Figure 2.19: Line pixel lookahead [1]

utilize spatial locality in the vertical direction, and looks ahead before decoding the next line of pixels [1]. Not all the neighboring pixels should be kept in the internal memory for vertical direction locality, most pixels are following the vertical mode similarity. A reduced embedded SRAM stores the pixels in the above LCUs and the LPL scheme is to predict whether the pixels would be stored in to the SRAM or not. Most pixels are determined as a horizontal-related prediction in intra prediction of SKIP mode in deblocking filter. Two 2W-bit TAG buffers are required to record each prediction tag and perceive the contrast between Neighboring TAG and Decoding TAG. The deblocking filter and intra predictor generates corresponding TAG information to forecast whether the next line of pixel should be kept or not. To reduce the size of memory, the Figure2.19 is exploited to reduce the miss rate. One OR gate, five comparators, two multiplexers and inverter are used to implement [1]

2.5.3 DMA-like Buffer

The purpose of the DMA-like buffer is to store the correlated data in a MB in the external memory if they are not used immediately [2]. Using the DMA-like buffer, the internal memory can be reduced from 26K in 1080p resolution to only 0.5K bytes, which results in 98 percent reduction in memory size. For the dual external memories adopted in the design to reduce the required clock rate for memory access operations. With less than 10 percent increase in external memory bandwidth, the trade-off between internal memory and external memory bandwidth provides flexibility for system designers. However, in the Figure 2.20, about 83.4MB/s of external memory bandwidth occupies the 9.5 percent in total 878MB/s by using this technique. If the power consumption is calculated utilizing Micron Power Calculator, the result is 33.9mW, which is larger than [1]. As a result, it is not good to put the almost internal data off the chip.

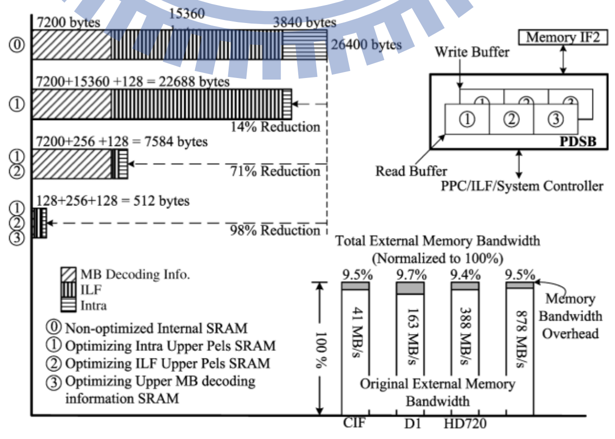

7200 bytes
15360 bytes
3840 bytes
26400 bytes
⓪
①
7200+15360 +128 = 22688 bytes
14% Reduction
①②
7200+256 +128 = 7584 bytes
71% Reduction
①②③
128+256+128 = 512 bytes
98% Reduction
MB Decoding Info.
ILF
Intra
⓪ Non-optimized Internal SRAM
① Optimizing Intra Upper Pels SRAM
② Optimizing ILF Upper Pels SRAM
③ Optimizing Upper MB decoding information SRAM
Memory IF2
Write Buffer
PDSB
Read Buffer
PPC/ILF/System Controller
Total External Memory Bandwidth (Normalized to 100%)
9.5%
9.7%
9.4%
9.5%
Memory Bandwidth Overhead
100 %
41 MB/s
163 MB/s
388 MB/s
878 MB/s
Original External Memory Bandwidth
CIF
D1
HD720

Figure 2.20: Memory reduction based on the DMA buffer [2]

2.6 Summary

Therefore, in the proposed algorithm, the memory would not totally cut into the off-chip memory. Also, the proposed I frame decoder is composed of the inverse discrete transform, intra predictor and in-loop filter which will be prototyped in FPGA platform and demo the video in the verification mechanism. Besides, in the next proposed algorithm, the proposed method can further achieve lower memory size with higher hit rate based on the memory hierarchy architecture compared with [1]. With the system pointview, the proposed method has better performance including memory access power consumption, external memory bandwidth and miss rate. The next chapter will describe the algorithm in detail.

Chapter 3

Proposed Algorithm

Due to the algorithm of HEVC, the on chip memory is required for the hardware to use. The huge on chip memory will cause the power consumption and waste the chip area efficiency. According to the conclusion summarized in section 2.6, that the traditional approaches are commonly using internal line buffers to store the un-filtered pixels for intra prediction and filtered pixels for in-loop filters.

3.1 Above Line Buffer Sharing

For system viewpoint, the intra predictor and deblocking filter in video codec are the memory dominated modules. In the hardware design concept, the correlated pixels need to be stored into the line buffers for waiting the next line of LCUs are available. For intra prediction, the reconstructed pixels stored into line buffer should be in the previous in-loop filter stage. The deblocking filter utilizes 4-line buffers to store the pixels which are filtered in vertical edge only as shown in Figure 3.1. The pixels would be used since the horizontal edge is started for the next new line LCUs are available. Therefore, the notification of the memory accesses would be impact to the hardware design. The concept of reducing internal memory in above line buffer sharing is to schedule the data path between intra prediction and deblocking filter. In Figure 3.2, the 4-line buffer is composed of two parts, one is original line buffer which saves the reconstructed pixels for the intra predictor to use, and the other parts 3-line buffer is saving the vertical edge filtered pixels for the deblocking filter to use. The details of data scheduling will be described in the Algorithm 1 below.

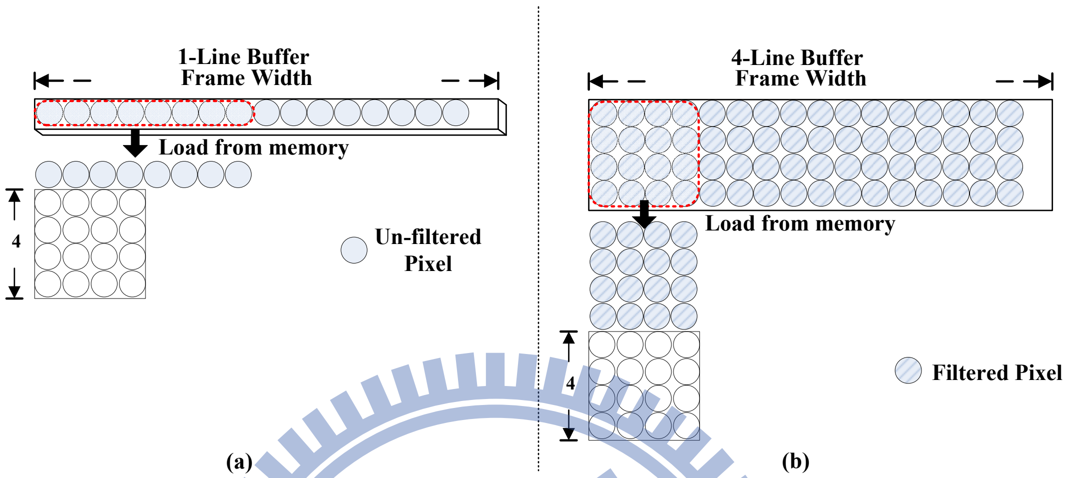


Figure 3.1: Intra and De-blocking Line Buffers

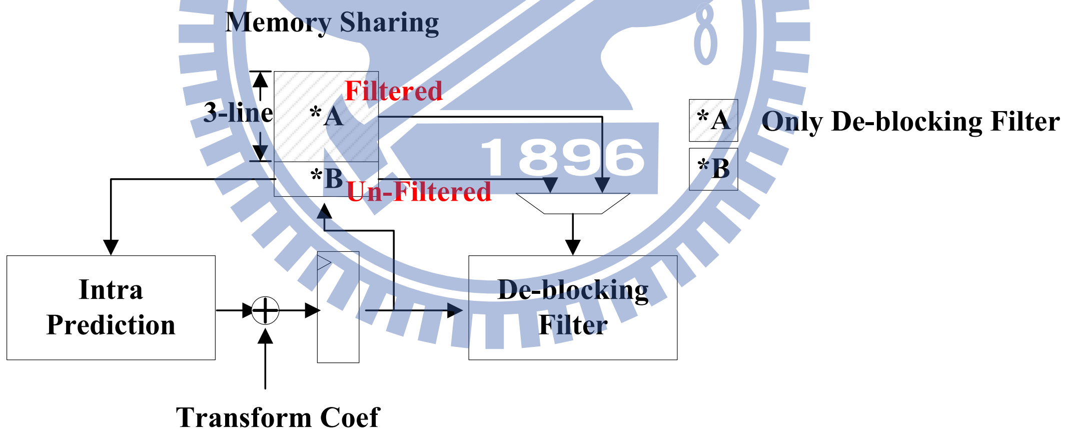


Figure 3.2: Intra and De-blocking Line Buffers Sharing

Assume the block 0 is firstly in intra prediction stage, it loads the reconstructed pixels which are un-filtered from the *B line buffer as shown in Figure 3.3. The pixels should be stored into registers in order for the deblocking filter to use. After finishing the intra prediction process, the predicted pixels will be added with transform residuals to become the reconstructed pixels and then will be stored into the same address in the *B line buffer as shown in Figure 3.4. In the Figure3.5, the block 1 will do the same intra prediction process, the block 0 will begin to

Algorithm 1: Shared Above Line Buffer

```
   Input: $Intra\ Info$ and $Deblocking\ Info$
   Output: $Deblocked\ Pixels$
1  forall the $MB_{No}$ such that $j \leq Width/16$ do
2      Intra Prediction:
3      step 1: Store bottom reconstruced pixels to 1-line buffer$_{intra}$
4      Deblocking Filter:
5      step 1: Store bottom 3 line of vertical filtered pixels to 3-line buffer$_{df}$
6  end
7  $Next\ Line\ Pixels\ Available$
8  forall the $MB_{No}$ such that $j \leq Width/16$ do
9      Intra Prediction:
10     step 1: Load reconstruced pixels from 1-line buffer$_{intra}$
11     step 2: Store bottom reconstruced pixels to 1-line buffer$_{intra}$
12     Deblocking Filter:
13     step 1: Load 3 line of vertical filtered pixels from 3-line buffer$_{df}$
14     step 2: Load reconstruced pixels from 1-line buffer$_{intra}$
15     step 3: Data recovery for the reconstructed pixels
16     step 4: Store bottom 3 line of vertical filtered pixels to 3-line buffer$_{df}$
17 end
```

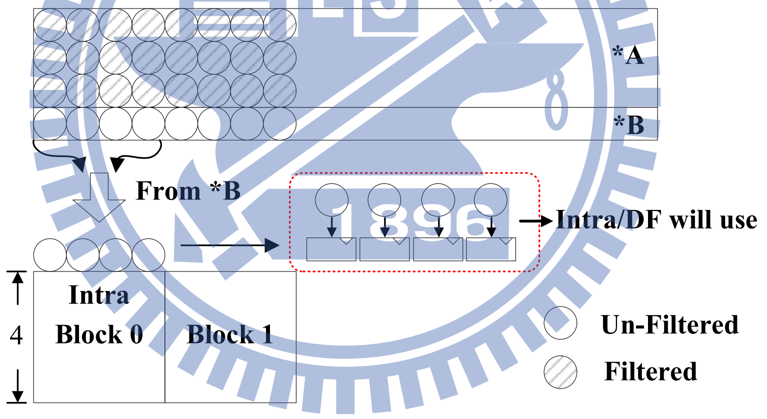


Figure 3.3: Intra and De-blocking Line Buffers Sharing

deblock. The above line pixels which are being loaded from the 3-line buffer *A and registers are the input for the deblocking filter. After finishing the deblocking filter, the 12 pixels with red dotted-circle in 4x4 block would be stored back into 3-line buffer *A as shown in Figure 3.6. The advantage of the line buffer sharing is that original 5-line buffers will be decreased to the 4-line buffer about nearly 20% reduction.

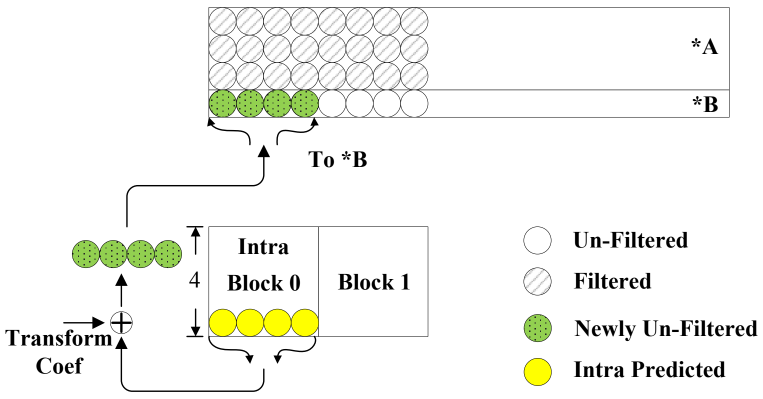


Figure 3.4: Intra and De-blocking Line Buffers Sharing

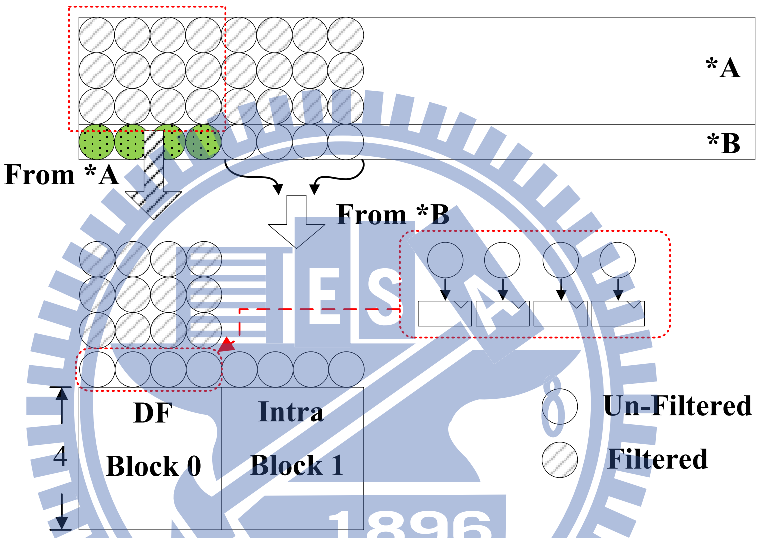


Figure 3.5: Intra and De-blocking Line Buffers Sharing

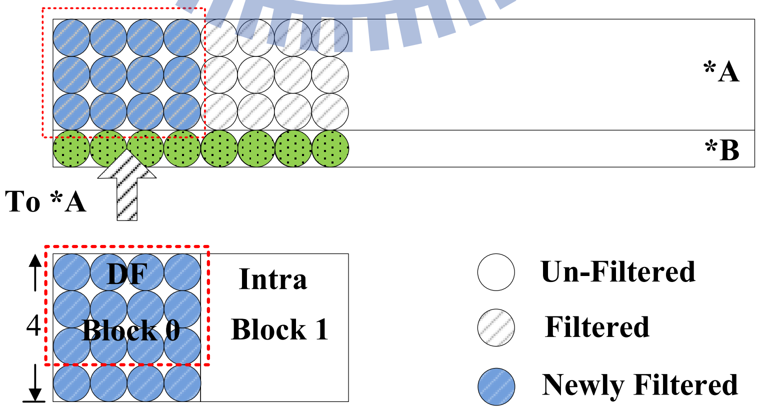


Figure 3.6: Intra and De-blocking Line Buffers Sharing

3.2 Data Recovery

In the deblocking filter process in the HM-7.1 reference software, the filtering order is following the vertical edge first in the frame-level, and then horizontal edge is the last as shown in Figure 3.7. However, in the hardware design, the frame-level is not possibly to be implemented for the silicon area concern. As a result, the 16x16 macroblock based filtering is the hardware trend. In the 16x16 macroblock-level, the right-most 4 4x4 blocks need to be stored into registers in order for the next right 16x16 macroblock, while the bottom 4 4x4 blocks need to store into 3-line buffers for the next boundary 16x16 macrblock as shown in Figure 3.8.

As a result, in the line buffer sharing, the 12 pixels loaded from the 3-line buffer *A plus the 4 pixels loaded from the 1-line buffer *B which are not through the vertical edge filter should be filtered again to make sure the data are valid. As shown in Figure 3.9, the pixels should be in vertical edge filtering in order to guarantee the value correctness. The disadvantage of line buffer is that before doing de-blocking filter, the data recovery should be done first, this will cause some clock cycles waste. In the table,

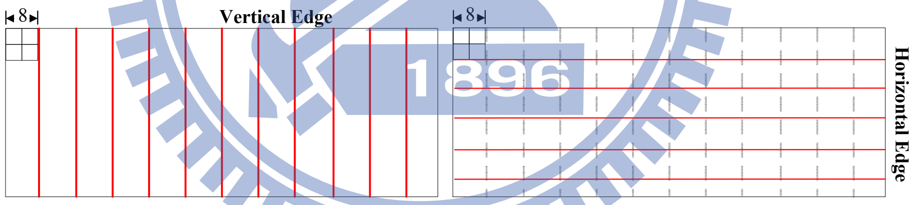


Figure 3.7: Deblocking filter in vertical edge and horizontal edge

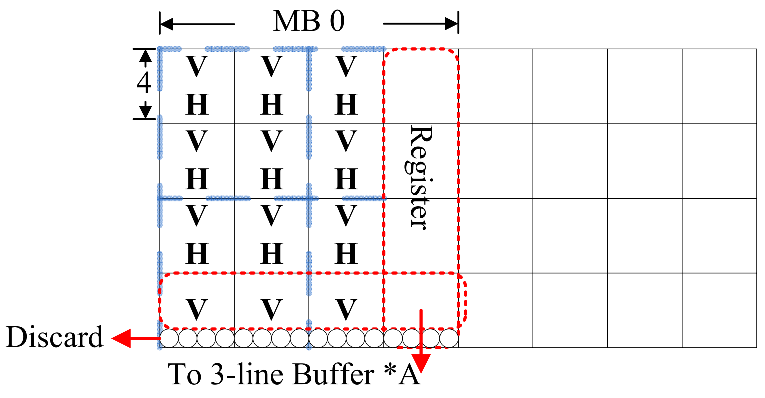


Figure 3.8: Memory sharing in deblocking filter

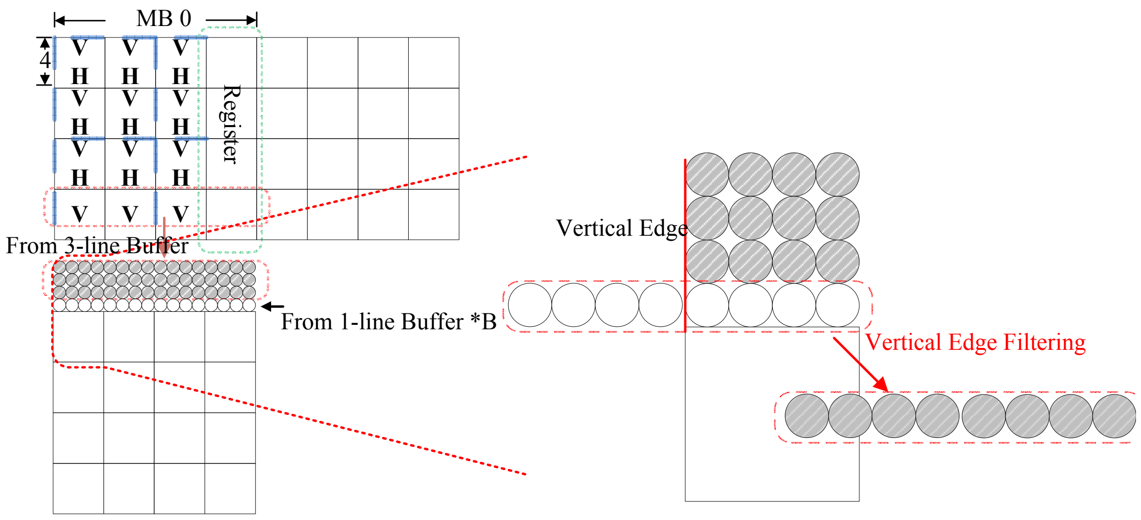


Figure 3.9: Memory sharing with data recovery

3.3 Prediction-base with Memory Hierarchy

The in-loop filter is the most memory-dominated in the video decoder. Therefore, the effort to reduce the internal memory for in-loop filter is worth doing it. To adopt the memory hierarchy shown in Figure 3.10, the purpose is to reduce the deblocking filter 3-line buffer SRAM size for saving the power consumption. The second level memory hierarchy is to exploit the high data correlation which would be used often in the decoding process. With the support of the smaller SRAMs with size decides by the designer, the pixels could be first stored and would reduce the accessing from the external memory. In the proposed method, we exploit the memory hierarchy by utilizing the spatial locality of neighboring pixels to reduce the line buffer size and the access of the external memory. In the reference software HM-7.1, the deblocking filter adopts the frame-level coding order. Therefore, in Figure 3.11, the edge detection process which is described in the previous section is available due to the pixels below the boundary edge in the Q block are available. However, in the hardware design, the frame-level coding is not the perfect choice to follow due to the huge hardware buffer cost. As a result, if we adopt the 16x16 macroblock pipeline, the pixels below the boundary edge are un-available in the Q block. With the unknown pixels below the boundary, the filter edge detection could not accurately judge the edge whether it is over-sharp or smoothed.

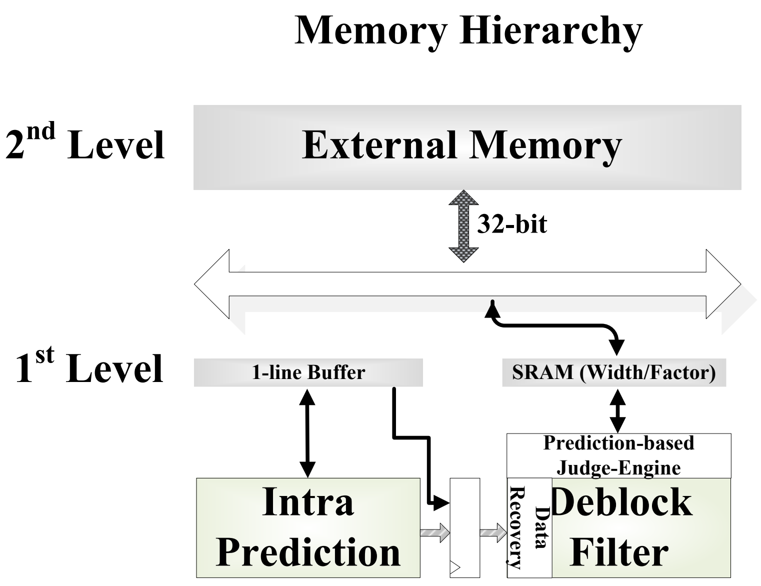


Figure 3.10: Memory hierarchy stage

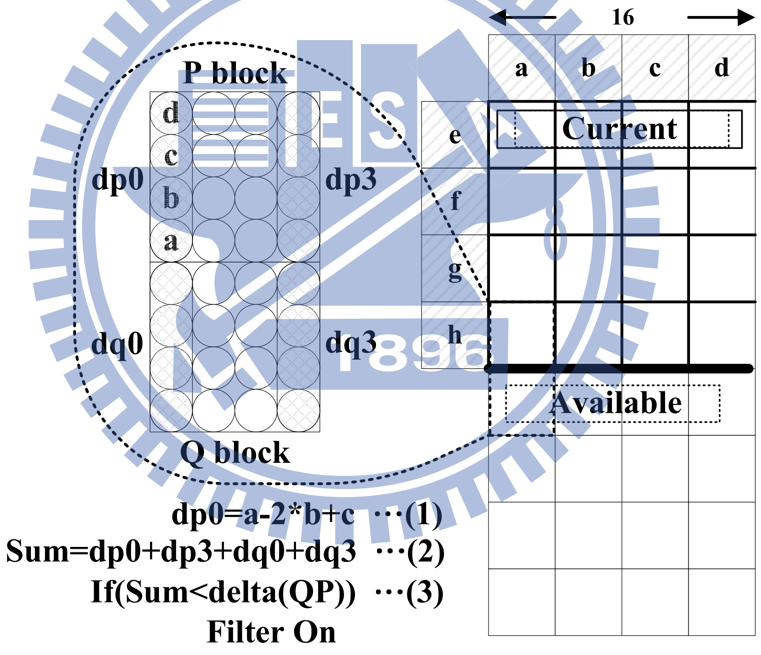
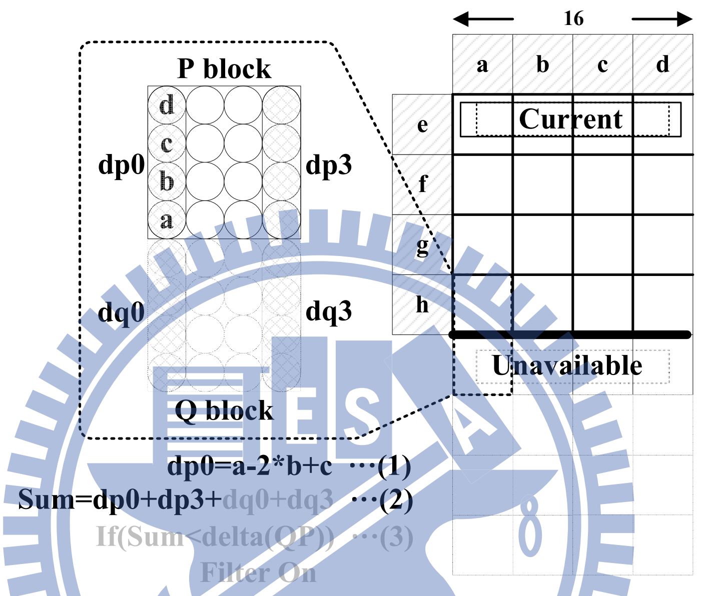
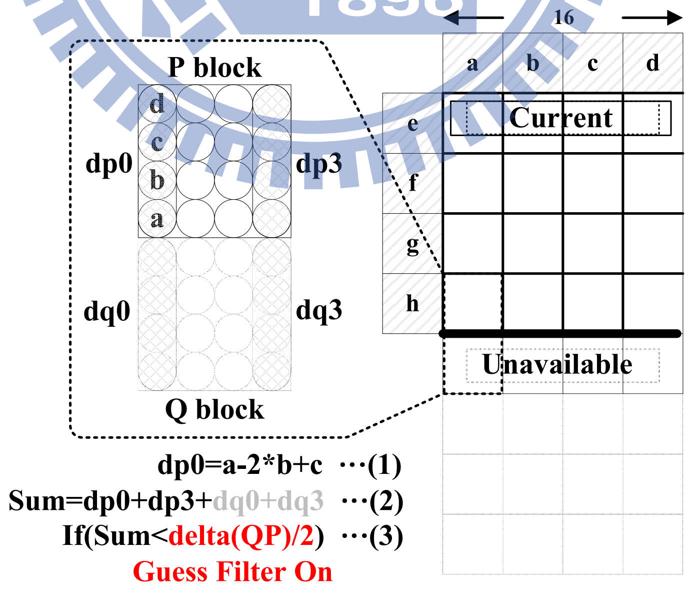


Figure 3.11: Deblocking filter in detection mode

To pre-judge whether the filter edge open or not, we proposed $Prediction - based\ Judge\ Engine$ to foresee the detection results. In Figure 3.12(a), although the pixels are un-available, the equation 2 in the Figure 3.12(a) can only get two parameters dp0 and dp3. We can still acquire the sum of dp0 and dp3. Assume that the spatial locality in the neighboring blocks, the dp0 will be highly similarly to the dq0. Therefore, the prediction of the filter detection can be implemented as equation 3 as shown in Figure 3.12(b), that we assume if the sum of dp0 and dp3 is smaller than the delta divided by 2, then the filter is opened. Further, to reduce the SRAM size, the judge engine can be exploited to detect whether the vertical filtered pixels would be used for the next new line of LCUs or just un-used. The judge engine will forecast the line pixels would be used for the deblocking filter to decide whether the pixels should be stored into the limited SRAM or just leave them in to the external memory.



(a) Deblocking filter in edge detection with hardware



(b) Deblocking filter in edge detection with prediction

Figure 3.12: Deblocking Filter in Detection Mode with Prediction

3.4 Summary

In the conclusion of the proposed algorithm including the shared above line buffer and the memory hierarchy, the SRAM distribution of the total in the video decoder is highlighted in Figure 3.13. In the Full HD resolution, the frame width is 1920-pixel long. As a result, in the traditional approaches, the decoder requires 9.5 Kbytes. With the shared above line buffer, 20% reduction could be achieved. Furthermore, with the reduction factor is set as 8. The reduction could further achieve totally 85% in the system.

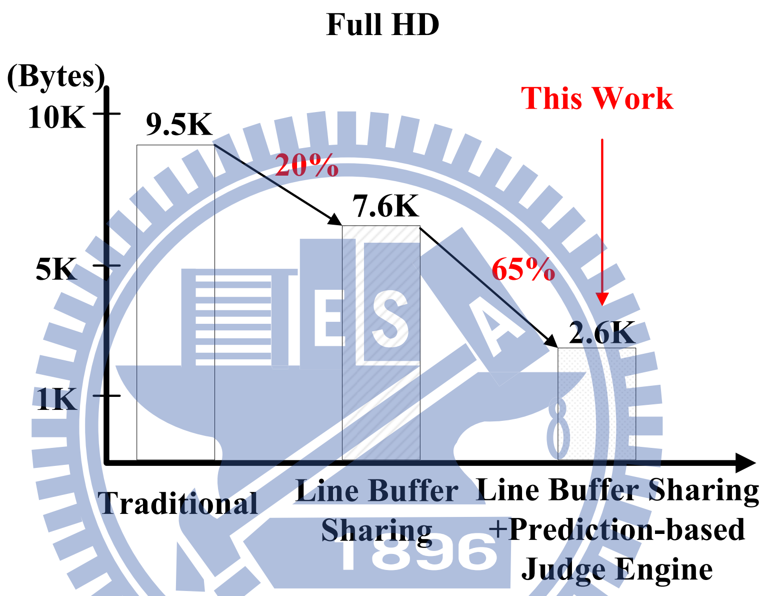


Figure 3.13: Memory Comparison

Chapter 4

Proposed Architecture

This chapter describes the proposed architecture with the line buffer sharing and memory hierarchy. Besides, the main component of the proposed I frame decoder containing the inverse transform, intra prediction, and de-blocking filter and adaptive loop filter will also be further described. Meanwhile, to meet the super-vision TV standard 8Kx4K, instead of speeding up the running frequency because of high power consumption, the wavefront parallel processing architecture will also be mensioned.

4.1 Design Target

In Figure 4.1 shown, that the video decoder target is aimed at higher resolution, for the real-time decode Full-HD [11], the architecture adopts pixel-level parallelism in each component to meet the Full-HD specification. Moreover, in recent years, the video decoder is targeted at multiview decoding, 3D-TV resolution, and super-vision TV applications. There is no doubt that only appling the pixel-level parallelism could not support the above applications. Therefore, the module-level parallelism is popular nowadays to meet the real-time decode in Ultra-HD resolution. In [12]and [13], they apply double entropy decoders and four entropy decoders respectively to further process more pixels in each pipeline stage.

For the beginning, the throughput of the I-Frame decoder is targeted at the super-high vision TV standard 8Kx4K at 30fps.

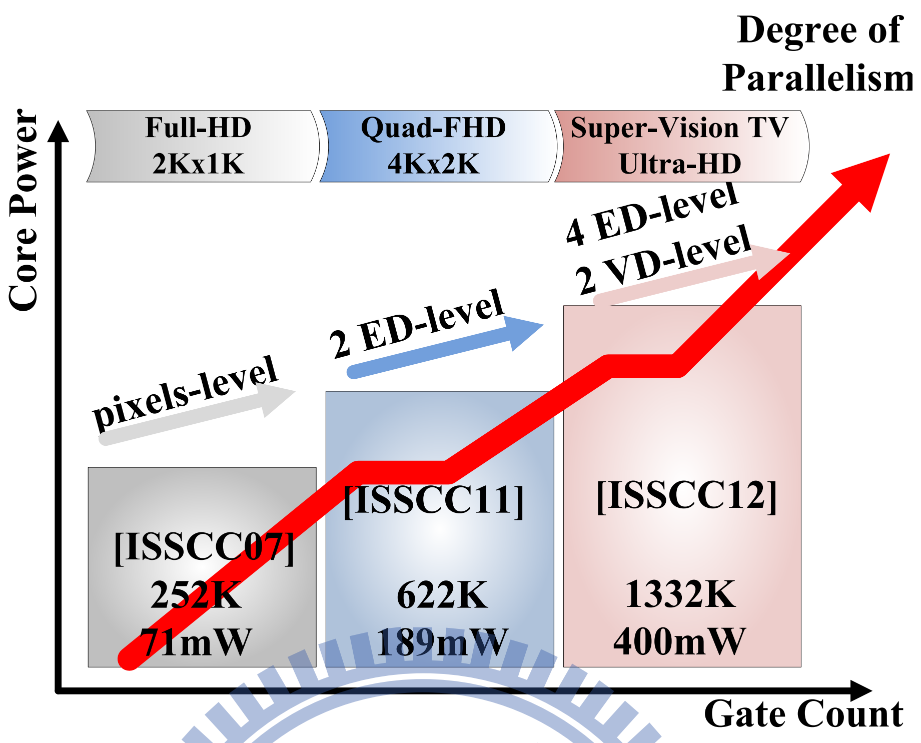


Figure 4.1: Higher throughput requires higher parallelism

Table 4.1: Pipeline stages vs throughput/frequency

	ISCAS '05 [11]	ISCAS '05 [14]	ISSCC '07 [2]	VLSI '09 [15]	ISSCC '10 [16]	JSSC '11 [12]	ISSCC '12 [13]	ISSCC '13 [17]
Throu-ghput	2048x 1024 @30fps	1920x 1080 @30fps	1920x 1080 @30fps	1920x 1080 @30fps	4320x 2160 @24fps	4320x 2160 @60fps	7680x 4320 @60fps	3840x 2160 @30fps
Frequ-ency	120MHz	100MHz	120MHz	200MHz	210MHz	175MHz	400MHz	200MHz
Cycles /16x16	488	411	494	411	240	80	102	205

The operation cycles for each stage should be enough to meet the 8Kx4K resolution. In Table 4.1, the list are the video decoders throughput versus frequency and stage cycles for each 16x16 block. In the [11], [14], [2], [15], the processing cycles in pipeline stage are about 400 500 cycles. However, if the frequency is fixed, the cycles would be divided by 16 for targeting 8Kx4K resolution. Limited cycles would be hard enough for the hardware implementation. Consequently, the technique of the parallelism is utilized in the [12], [13] to support the higher resolution.

In the system design of the I-Frame decoder, the target of architecture is to achieve real-time decode in 8Kx4K efficiently. Certainly, the wavefront parallel processing (WPP) in HEVC is used to further enhance the coding throughput by utlizing parallel hardware architecture. In the mean time, the hardware pipeline stages are decided as input buffer, inverse transform, intra prediction, de-blocking filter, adaptive loop filter and finally the write out stage as shown in Figure 4.2. With the support of the WPP, the decoder adopts the 4-parallel hardware to achieve 1Gpixels/sec as shown in Figure 4.3. As a result, the I-frame decoder with 4-parallelism wavefront parallel processing is proposed to enhance our decoder throughput.

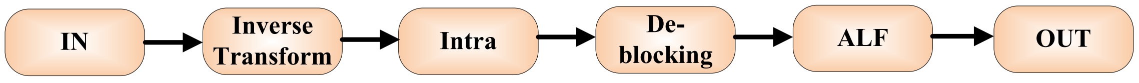


Figure 4.2: 6 pipeline stages

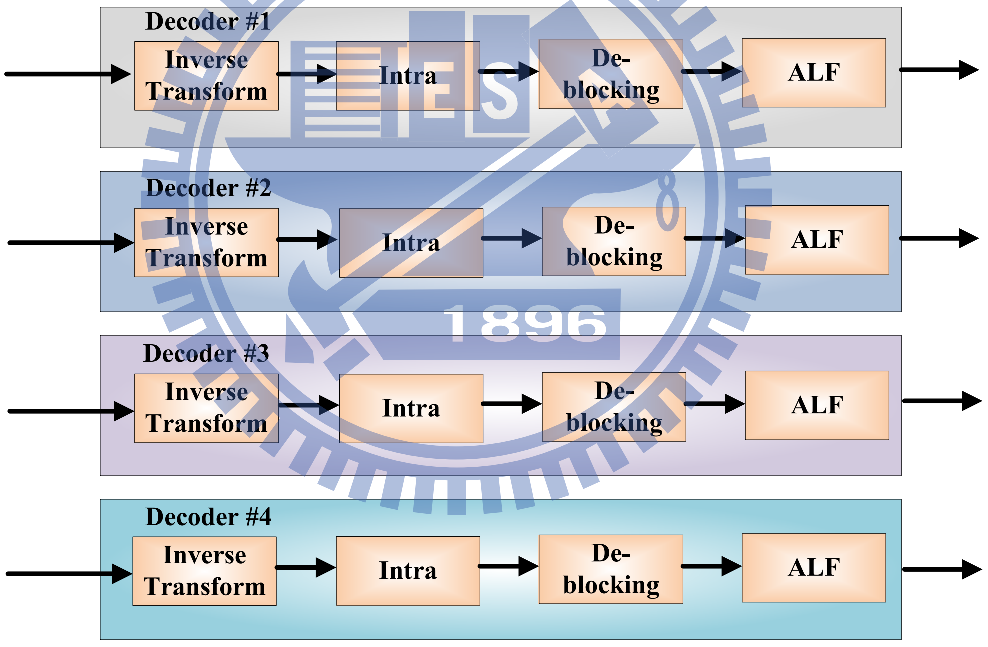


Figure 4.3: 4 parallelism decoder to enhance throughput

4.2 Inverse Transform Coder

The high data rate 2-Dimensional inverse transform coder is proposed to accelerate the pixel throughput for HEVC system. Certainly, the hardware can suitably arrange the data scheduling with up to 16 pixels/cycle with parallel processing engines in each 4x4 or 8x8 inverse discrete cosine transform coder. The sections below will describe details of the transform core.

4.2.1 Hardware Sharing

The proposed inverse discrete cosine and sine transform hardware is based on the HEVC algorithm. As shown in Figure 4.4 and Figure 4.5, the core of processing engine contains 8 multipliers and adders/subtractors in discrete cosine transform, while the core of discrete sine transform contains 9 multipliers and 7 adders/subtractors. The rounding is implemented easily by using wire shifting. Moreover, each processing engine can process 4 residuals in two cycles.

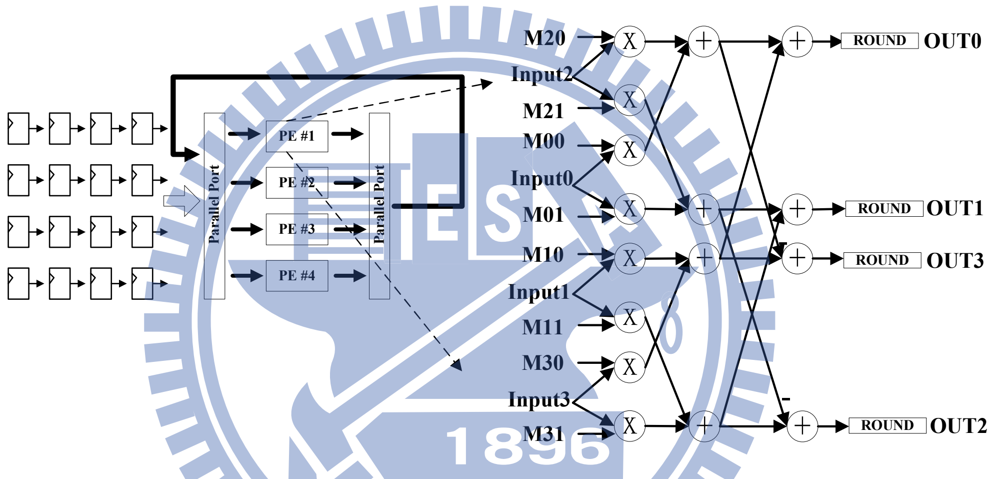


Figure 4.4: 4x4 inverse discrete cosine transform coder

Therefore, the 4 numbers of processing engine can fluently process a 16x16 coefficient block with 4x4 mode given by the entropy decoderin 16 cycles.

In Figure 4.6, the processing engine of the 8x8 transform coder can share the 4x4 processing engine to reduce the hardware cost with about

Finally, the 4x4 and 8x8 inverse discrete cosine and sine transform coders would be integrated in the I-Frame decoder. With 90nm CMOS technology, the system running frequency is 320MHz which can achieve 5G pixels/sec pixel throughput with the gates. As shown in Table 4.2, the throughput of the transform coder are listed the [18], [19], [20], [21] and [22] below. As the frequency is normalized, the proposed transform coder can achieve high throughput 2x 15x times higher.

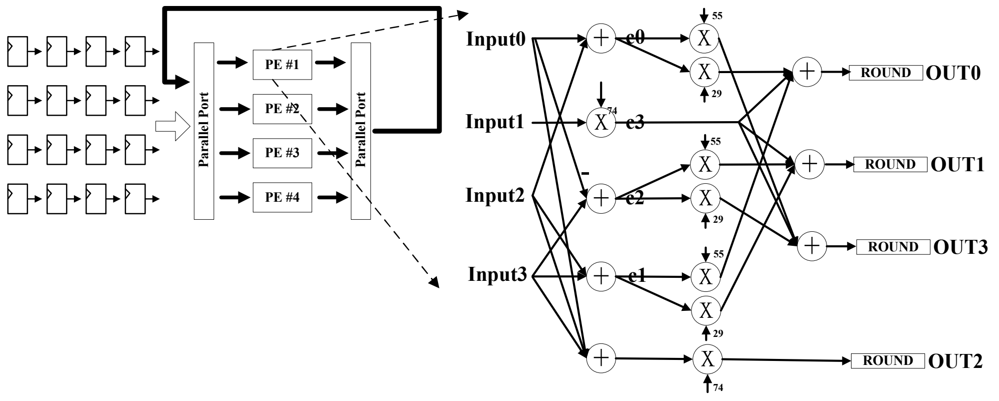


Figure 4.5: 4x4 inverse discrete sine transform coder

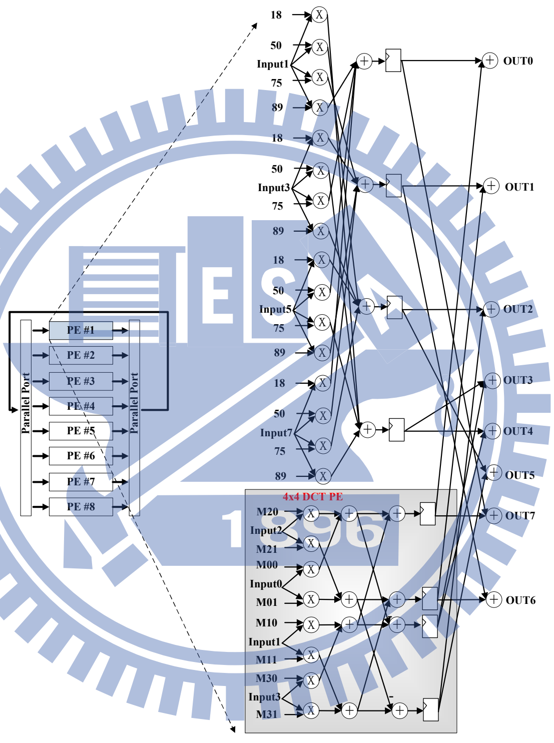


Figure 4.6: 8x8 inverse discrete sine transform coder

Table 4.2: Pixel throughput in different designs

	CSVT '06 [18]	TVLSI '08 [19]	ISCAS '09 [20]	ISOCC '10 [21]	TVLSI '12 [22]	Proposed
Throughput (pixels/sec)	800M	100M	149M	800M	167M	5G
Frequency	100MHz	100MHz	149MHz	100MHz	167MHz	320MHz

4.2.2 Time Scheduling for WPP

When it comes to the 4 parallelism of decoder, the high data throughput of the tranform coder can output enough residual data in 1 pipeline stage for the 4 intra predictors and in-loop filters as shown in Figure 4.7. Therefore, the hardware cost can be reduced from the original 4 numbers of transform coders to the 1 as shown in Figure 4.8.

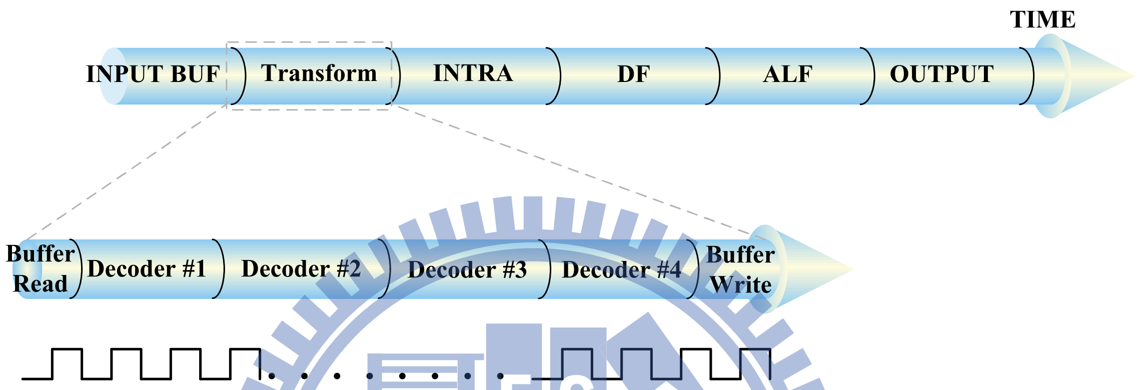


Figure 4.7: Transform scheduling in WPP

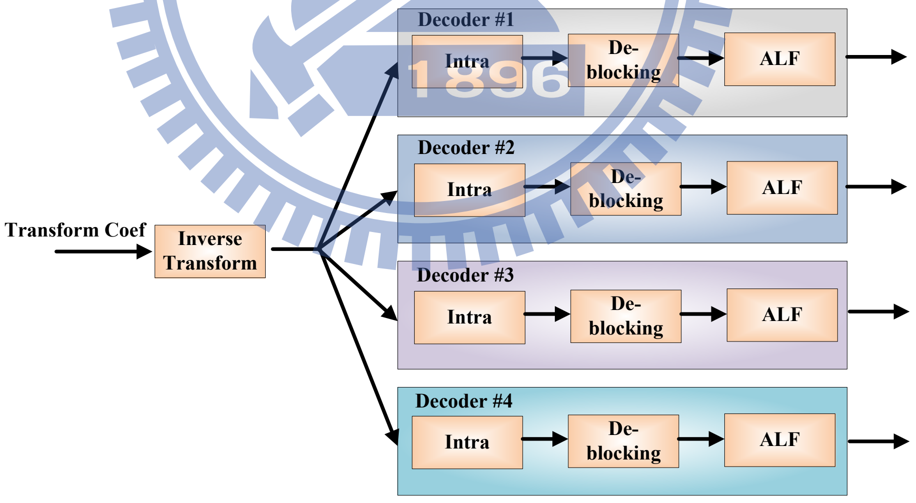


Figure 4.8: System design of the WPP architecture

4.3 Intra Prediction

This section is talking about the architecture design of the intra predictor for HEVC standard. Figure 4.9 shown in here is intra predictor architecture with 1-line buffer for Ultra-HD resolution. We partition 3 parts, one is reference sample selection, another is intra filter processing, the other is write to buffer.

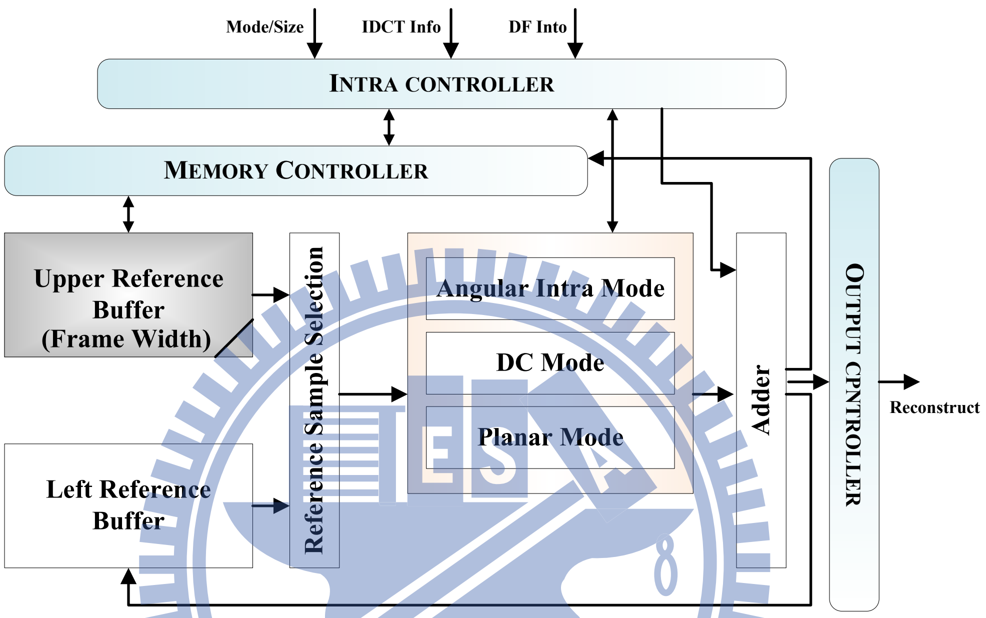


Figure 4.9: Architecture design of the intra prediction

4.3.1 Reference Sample Selection

In the intra prediction, due to the limited external memory bandwidth, the required upper reference buffer is used to temporily store the un-filtered pixels after the reconstruction with residual data. As shown in Figure 4.10 , the unfiltered pixels stored in the upper reference buffer which are fetched by the 16x16 block are used for the top 4 4x4 blocks, and the left reference buffer is storing the previous block of the right-most pixels. When the mode ans size information are coming , the reference sample selection will choose what pixels in the upper and left reference buffer would be used based on the intra mode ans intra size. As shown in Figure 4.11, if the mode belongs to the upper direction, then the samples would choose the A. Similarly, the mode is DC or upper-left-corner direction, then B is chosen as the samples. Last, the range of C are chosen for the left mode directions.

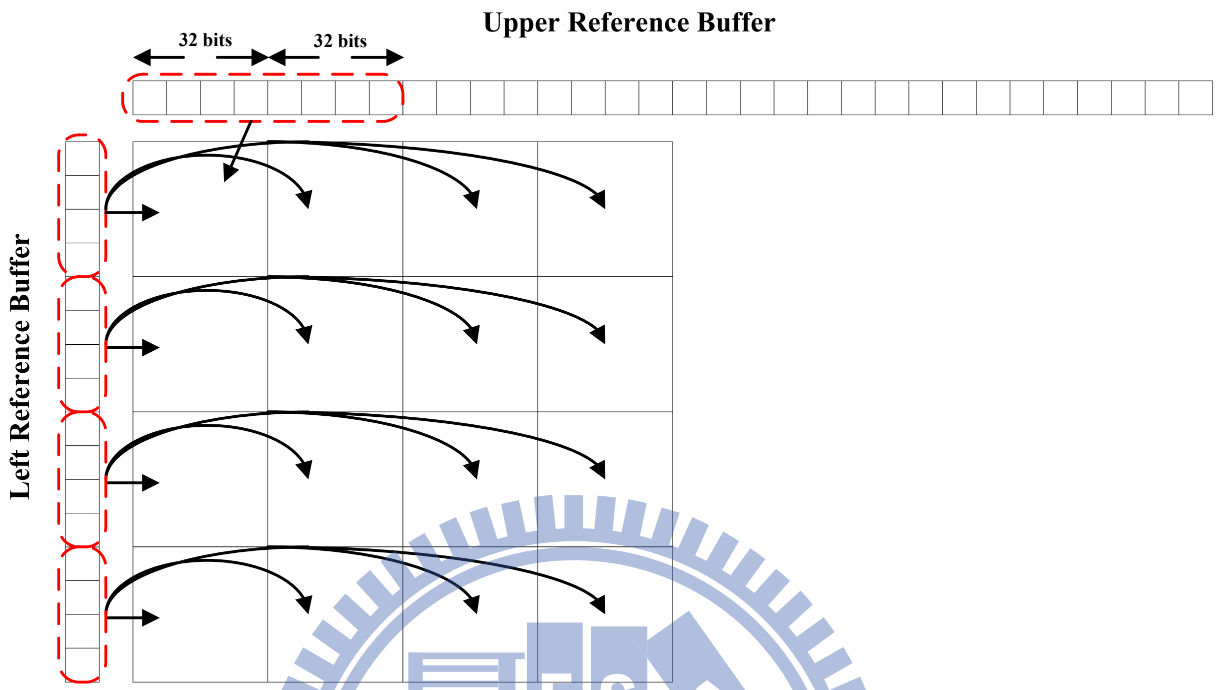


Figure 4.10: Reference sample read of the intra prediction

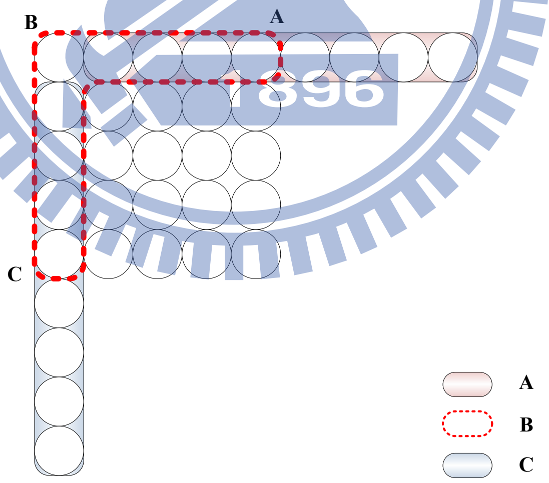


Figure 4.11: Reference sample range of the intra prediction

4.3.2 Angular Intra Mode

As for the angular intra prediction, the angle is defined as the displacement of the current pixel and top reference pixel in the vertical prediction. Also, it is defined as the right current pixel and left column pixel in the horizontal prediction. By using linear interpolation filter, the predicted pixels can be generated by choosing appropriate reference samples. The hardware challenge is the long operation cycles and how to design a reconfigurable intra predictor to handle all of the angular intra mode. As shown in Figure 4.12, the reconfigurable hardware design is proposed. In the first part, the intra mode would first convert into angles and then calculate the position (POSn, n=0$\tilde{3}$)between two input reference samples. During the positive

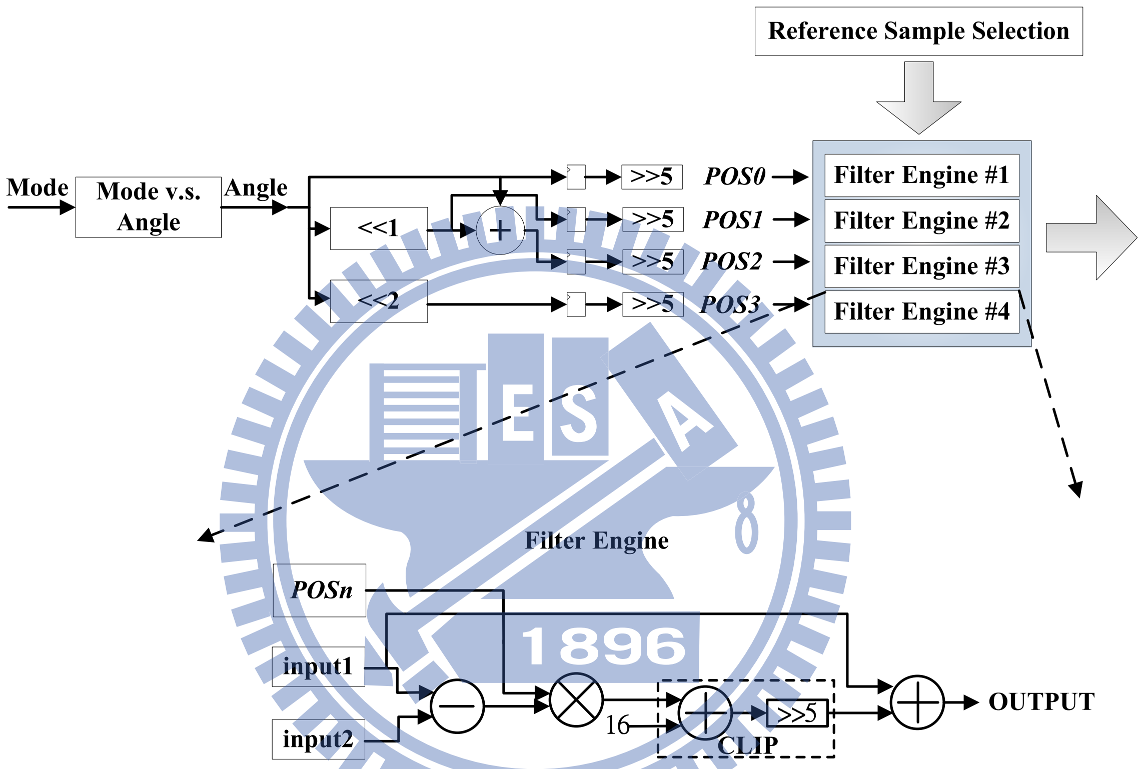


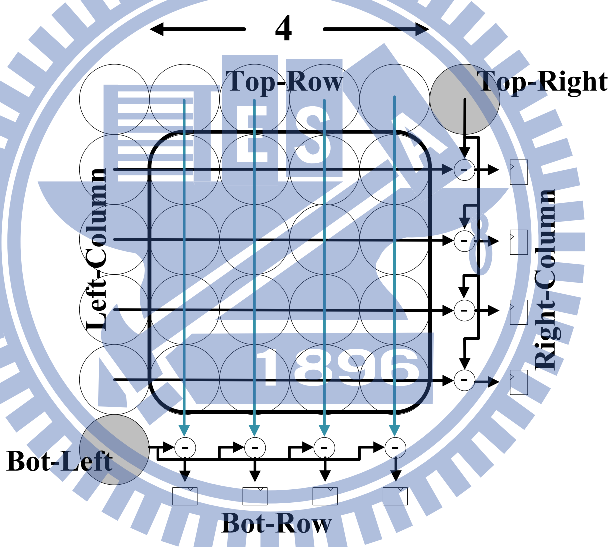
Figure 4.12: Filter engine of the intra prediction

angles or negative angles, the reference samples are divided by main and side array. As the vertical mode is chosen, the postive angle will demand the top reference samples as the main array for prediction. Similarly, the horizontal mode in the positive angle will also demand the left references as the same. Instead, for the negative angle, the projection of the side array into the main array is the first step. In addition, the lookup table and add operation are to select the needed samples for the prediction. For enhancing the prediction throughput to meet the 8Kx4K resolution requirement , we adopt 4-pixel parallelism to achieve 4 predicted pixels each cycle. The Filter engine which consists of 3 adder/subtractor and 1 multiplier is to interpolate

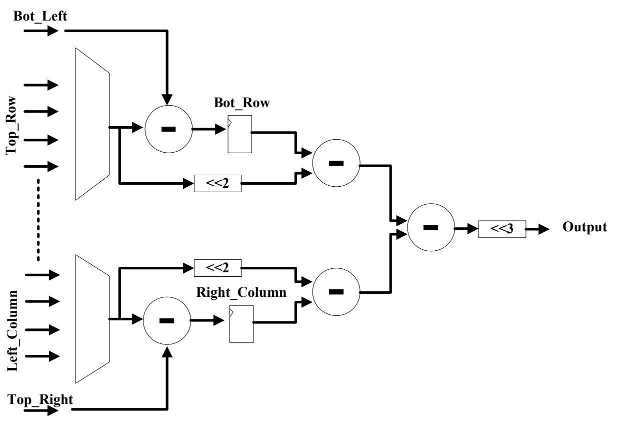
32-tap filter with 2 reference samples. The position (POSn, n=0$\tilde{3}$)between two input reference samples is the displacement calculated in the addition and shift step. Finally, the clip step is implemented only by wire-shifting.

4.3.3 Planar Mode

When the block is coded as planar mode, Figure 4.13(a) depicts that the right column values are produced by eliminating the left column and top right pixels. Also, the bot row values are produced by eliminating the top row and bottom left pixels. Moreover, Figure 4.13(b), the linear interpolation is implemented by the 3 subtractors with bottom row and the shifted top row pixels, the left column and the top right are the same, respectively. This approach in the



(a) Generate the bottom row and right column values



(b) Architecture Design of the Planar Mode

Figure 4.13: Planar Mode in Intra Prediction

intra prediction makes the pixel values continuous at the block edges. Therefore, the chances that applying the de-blocking filter to smooth the block effect are unusual. The technique of the planar mode improves the smooth surfaces, while the transform residuals added with predicted pixels would generate apparent blocking effect.

4.3.4 Write to Buffer

As the 16x16 block is coded 16 4x4 intra coded, the intra coding order is adopted $doublez$ scan order. Therefore, as shown in Figure 4.14, after the block 0 has been finished prediction and has been reconstructed with the residual data. The rightmost pixels are to be used for the

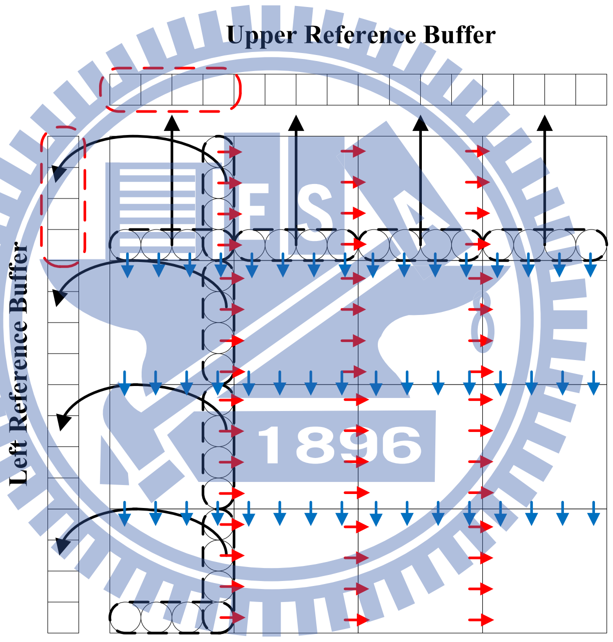


Figure 4.14: Write back operation of the intra prediction

right block 1, also, the bottommost pixels are to be stored for the below block 2. If the block 1 has finished, the rightmost pixels would need to store into the left reference buffer for the righ block 4, again, the bottommost pixels would be written to the upper reference buffer for the block 3 to use. When the block 2 begins, the left reference buffer are the previous block reconstructed pixels as usual. Due to the raster order, the reconstructed pixels should be saved on the block boundary in order for the incoming block to use. As a result, the blocks 10, 11, 14 and 15 should save the bottommost pixels to the internal SRAM.

4.4 De-blocking Filter

This section is talking about the architecture design of the de-blocking filter. As shown in Figure4.15, the de-blocking filter is partitioned into 4 parts, the edge detection unit, cache and left reference buffer, double transpose buffers, and filter unit. Due to the proposed algorithm,

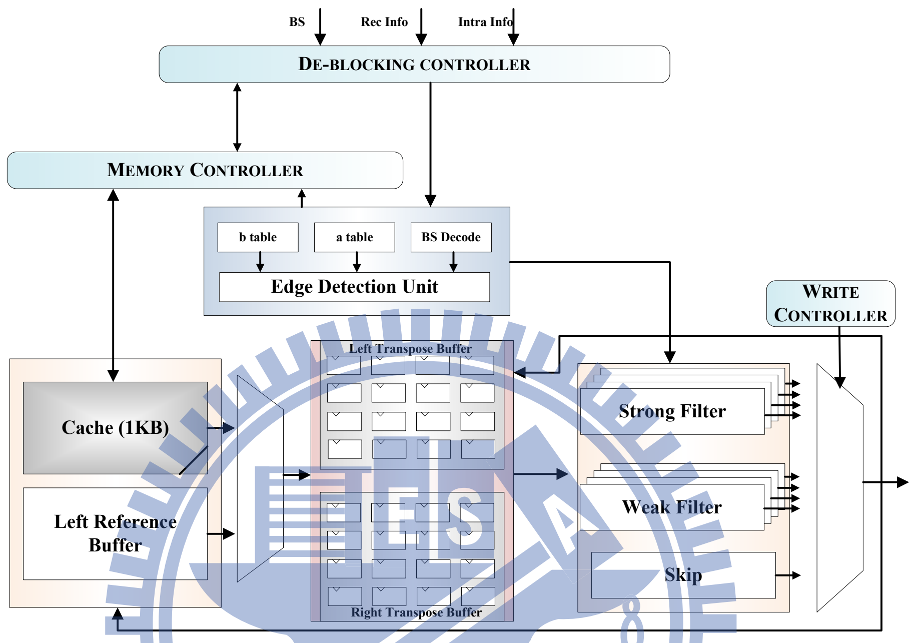


Figure 4.15: Architecture of the de-blocking filter

the 2nd memory hierarchy and line buffer sharing will be implemented in this section. Besides, the block filtering order is different from the previous standard H.264. The proposed order will also be described here.

4.4.1 Hybrid Filter Order

Deblocking filter should be considered in the memory access in terms of the filtering order. The architecture [6] which follows the specification's filtering order would produce repeated memory accesses, that the intermediate pixels have to be stored and fetched many times for the vertical and horizontal edge. Therefore, the memory accesses would be carefully considered for the operation cycle issue. Generally, the deblocking filter occupies $1/3$ computational complexity at the H.264 video decoder [23] Differentiate with H.264 standard, the filter operates 8x8

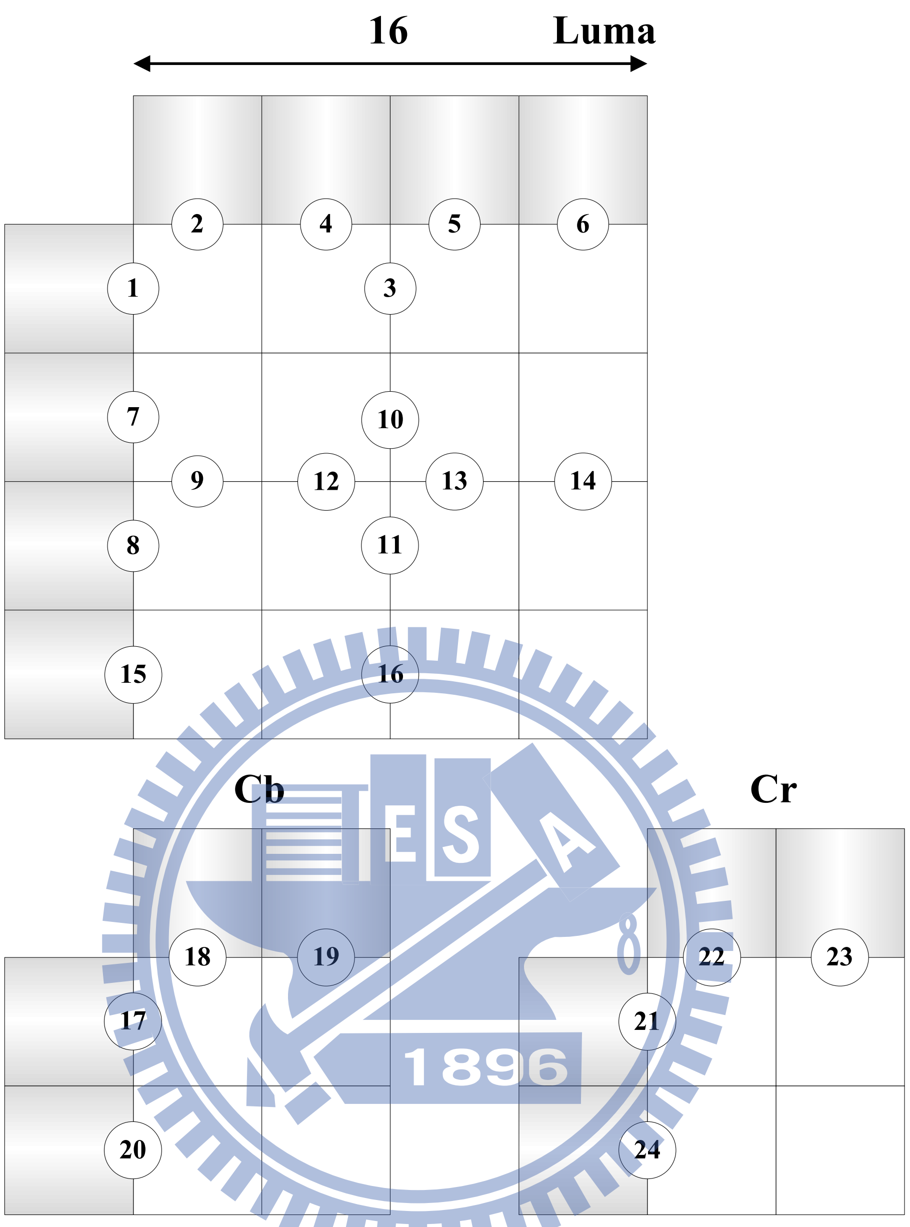


Figure 4.16: Hybird order of the de-blocking filter

boundaries instead of 4x4. In this sub section, we present the hybrid filter order still meeting the HEVC standard to reduce the internal SRAM access to further improve the access efficiency as shown in Figure 4.16. The vertical edges are filtered first, and then horizontal edges. The block is sized to 4x4 large. The numbers in the circle represents the processing order in the 4x4 filter unit. To solve the problem of the repeated memory accesses, the proposed hybrid scheduling could still meet the vertical edge first and then horizontal edge for following the pixel dependency.

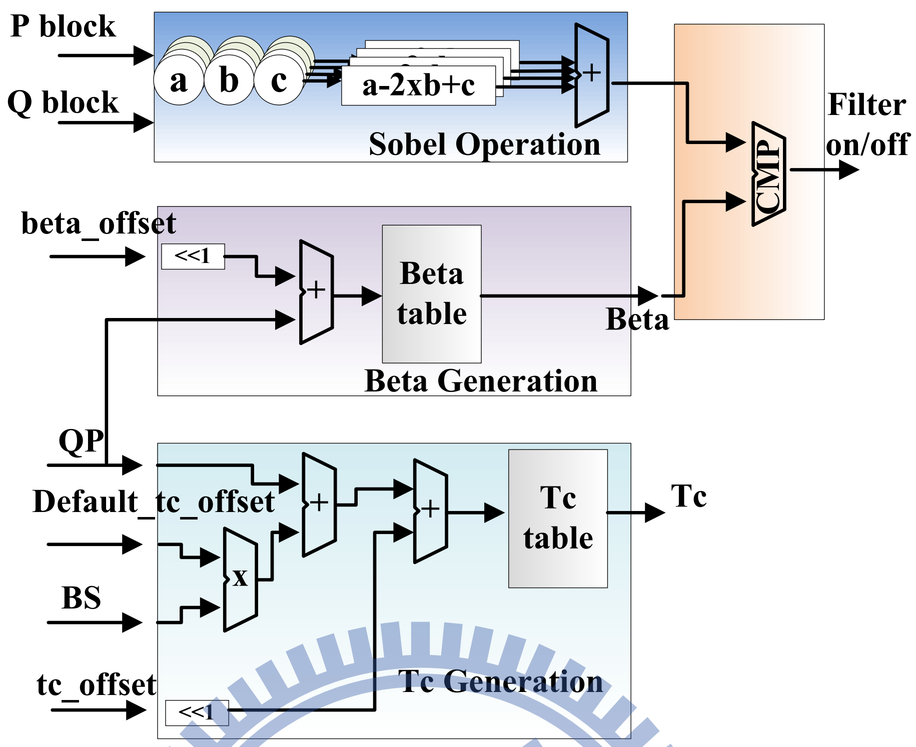


Figure 4.17: Edge detection of the de-blocking filter

4.4.2 Edge Detection Unit

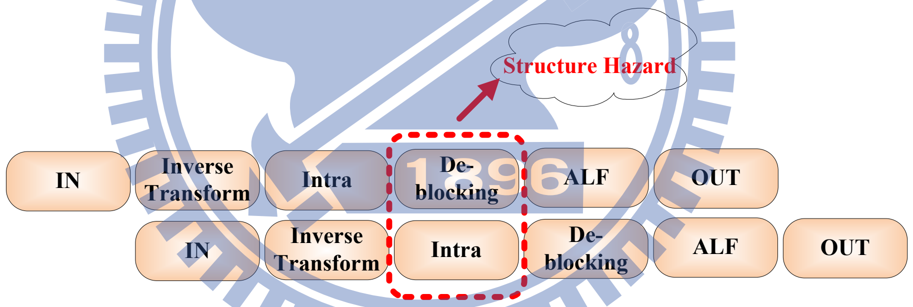
Applying the hybrid filter order, the edge detection unit is proposed to guarantee the edge whether over-sharped or not. As Figure 4.17 shows, the two parameters $beta$ and Tc, which are generated by utilizing the loopup table, quantization parameter, and also the boundary strength. The sobel operators are usually used in the edge detection of the image processing [24]. However, in the HEVC standard, the sobel operation is little different than other normal sobel operators. In the sobel operator (4.1), the coefficients of the x directions are applying the bilinear filter $[1, -2, 1]$ instead. Therefore, if the edge would be correctly checked using sobel operators, the double P and Q 4x4 block pixels should be send into the matrix multiplications. If the values are smaller than the $beta$, which means that the boundary is actually existed ans not need to be smoothed by appyling deblocking filter. However, if the values are bigger than the $beta$, the boundary is over-sharped, therefore, applying the deblocking filter to improve the video quality is important.

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ -2 & 0 & 0 & -2 \\ 1 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

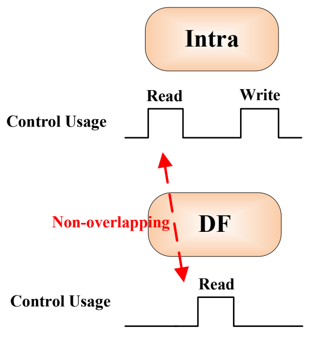
4.4.3 Memory Hierarchy and Buffer Sharing

1. Line Buffer Sharing

As shown in Figure 3.5, the unfiltered pixels could be loaded for the deblocking filter to use. While the intra coded block also could load the unfiltered pixels for intra prediction in the pipeline stage. To be considered for the SRAM usage, since the deblocking filter and intra prediction all need to load the SRAM at the same time, the un-avoided structure hazard would produce the problem as shown in Figure 4.18(a) Therefore, the data scheduling of the single port SRAM would be issued to solve the memory conflict.



(a) Structure Hazard of the Line Buffer Sharing



(b) Non-overlapping of the line buffer access

Figure 4.18: Structure Hazard of the Line Buffer Sharing

2. **Memory Hierarchy**

The proposed memory hierarchy reduces the internal SRAM for deblocking filter is very effective for achieving low power consumption. In [1], the power of the internal memory occupies about 70% of the video decoder. Due to the unnecessary storing all width of the pixels in the internal SRAMs for deblocking filter horizontal edge filtering, the proposed prediction-based cache mechanism is to eliminate the unusually-used pixels. Therefore, the prediction-based mechanism could contribute to the smaller cache size. The memory hierarchy will decide the high data-relation from the intermediate filtered pixels from the deblocking filter to the smaller cache.Therefore, with trade-off the external memory power consumption and the internal memory power, the analysis would be shown to present the approapriate cache size.

3. **Prediction-based Mechanism**

The prediction-based mechanism is proposed to utilize the edge detection unit in vertical direction and judge whether the boundary is to be filtered or not. It will decide whether the vertical filtered pixels would be saved into the cache in order to improve the access efficiency. The related pseudo code in 2 is shown that in the bottom edge of the 16x16 block, due to the unavailable pixels for the vertical direction Q block, the pre-edge detection could be operated to forsee the edge filter on or off. Currently, the sobel results may be half of the complete sobel results because of the spatial pixel locality. As the next line of pixels are available, the edge detection for the top edge and also can check the tag array to make sure hit or miss. In the next item, the miss rate analysis would be present to prove the proposed method would out-perform the previous design [1]. Besides, the architecture design of the prediction-based cache is shown in Figure 4.19.

4. **Performance Evaluation**

The proposed prediction-based mechanism with second memory hierarchy are applied to forsee whether the pixels would be stored into the internal SRAM or not. Increasing the internal memory size to store the intermediate pixels achieves lower external memory

Algorithm 2: Prediction-based Judgement

```
   Input: $Sobel\ Results$ and $beta$
   Output: $TAG$
1  forall the $j$ such that $j \leq Width/4$ do
2      Check Edge Detection Unit:
3      if Sobel Results<beta/2 then
4          Store to SRAM
5          $TAG[j] \leftarrow 1$
6      end
7  end
8  $Next\ Line\ Pixels\ Available$
9  forall the $j$ such that $j \leq Width/4$ do
10     Check $TAG[j]$:
11     if $TAG[j] = 1$ and Sobel Results<beta/2 then
12         HIT ←--
13     end
14 end
```

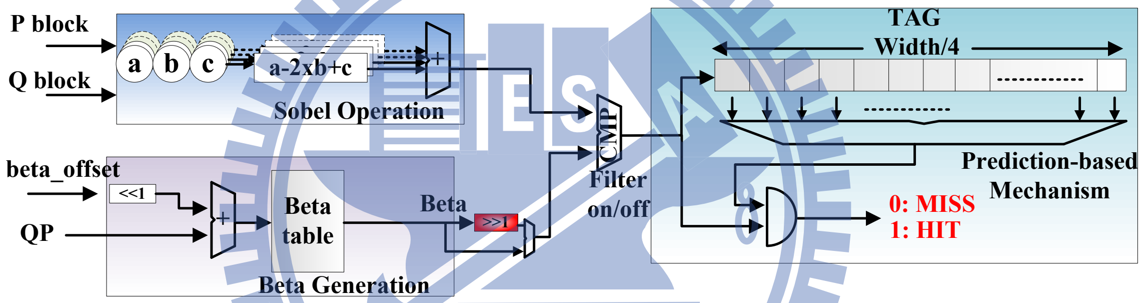


Figure 4.19: Prediction-based with memory hierarchy

bandwidth. However, the high hardware cost and full-speed SRAM power consumption will be a challenge for the video decoder. Therefore, to set the appropriate size of the SRAM, the analysis of the internal SRAM power and external memory bandwidth power consumption is important for the desicion. In [1], their work shows that in the factor of 8 is the compromisation with consideration of external SDRAM bandwidth, SDRAM power consumption and internal memory size. Due to the reduction factor of the internal SRAM, the penality of the miss rate would increase coming with external memory power consumption. As shown in Figure 4.20, the situation how the miss happens is that if the guess filter on or off array is 1, which means the pixels are stored in the SRAM if the size is still enough. However, if the the truth of the edge is on, but the top reference

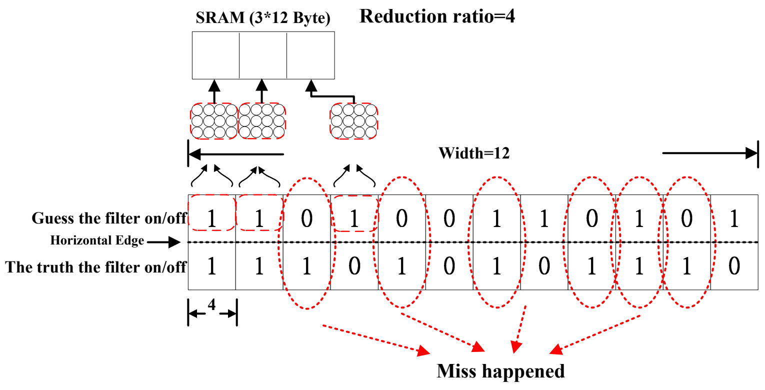


Figure 4.20: Miss rate analysis with memory reduction

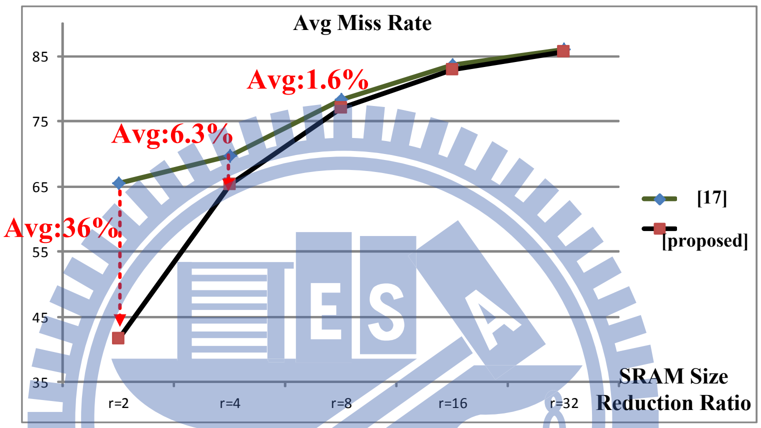


Figure 4.21: Miss rate analysis with memory reduction

sample pixels did not save in the internal SRAM, then the miss happens. Therefore, the signal will send into the external memory and load the pixels. In addition to comparing with [1], the proposed method can achieve better miss rate performance shown in Figure 4.21. With reduction factor of 2, the miss rate is reduded up to 36% average. As the factor doubles higher, the reduction is not apparently shown due to the limited memory size.

4.4.4 Strong/Weak Filter

In the last part of the deblocking filter is the filtering operation. It smoothes the over-sharp with discontinous edge. In this section, the proposed architecture of the strong and weak filter design would be present. In the deblocking filter, the filter engine would take 4 pixels on left and right side called P block and Q block. If the edge is strongly over-sharped, then the strong

Table 4.3: Deblocking filter cycle count

	ICME '03 [6]	ISCAS '05 [25]	ISSOC '10 [26]	Proposed
Cycle/MB	504	250	243	231
Gate	18.91K	19.64	N/A	54K

filter is chosen. The 3 modified output on each side would be changed during several adders interpolation, While, the weak filter is chosen, the modified 2 output on each side would be changed respectively shown in Figure 4.22. In the modified deblocking filter architecture, we adopt 4-filter parallelism respectively for strong and weak filter to enhance the coding speed. As shown in Figure 4.23, as the P and Q block all enter into the filter engine including strong and weak filter, the modified output would finally be chosen between the edge strength constraint. In the architecture of the strong filter, due the common items could be shared during the hardware resource, the area of the strong filter could be saved xx%. Also, the weak filter design composed of the delta generation depending on the 4 pixels on each side.

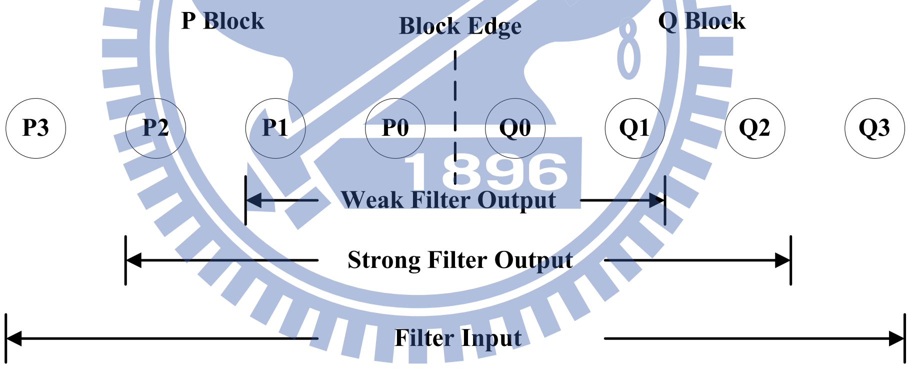


Figure 4.22: Input and output of the filter engine

4.4.5 Cycle Analysis

To sum up, the operational cycles during the deblocking stage is important to the pipeline stage decision. Therefore, in the Table 4.3, the comparisons with the operation cycles are listed below.

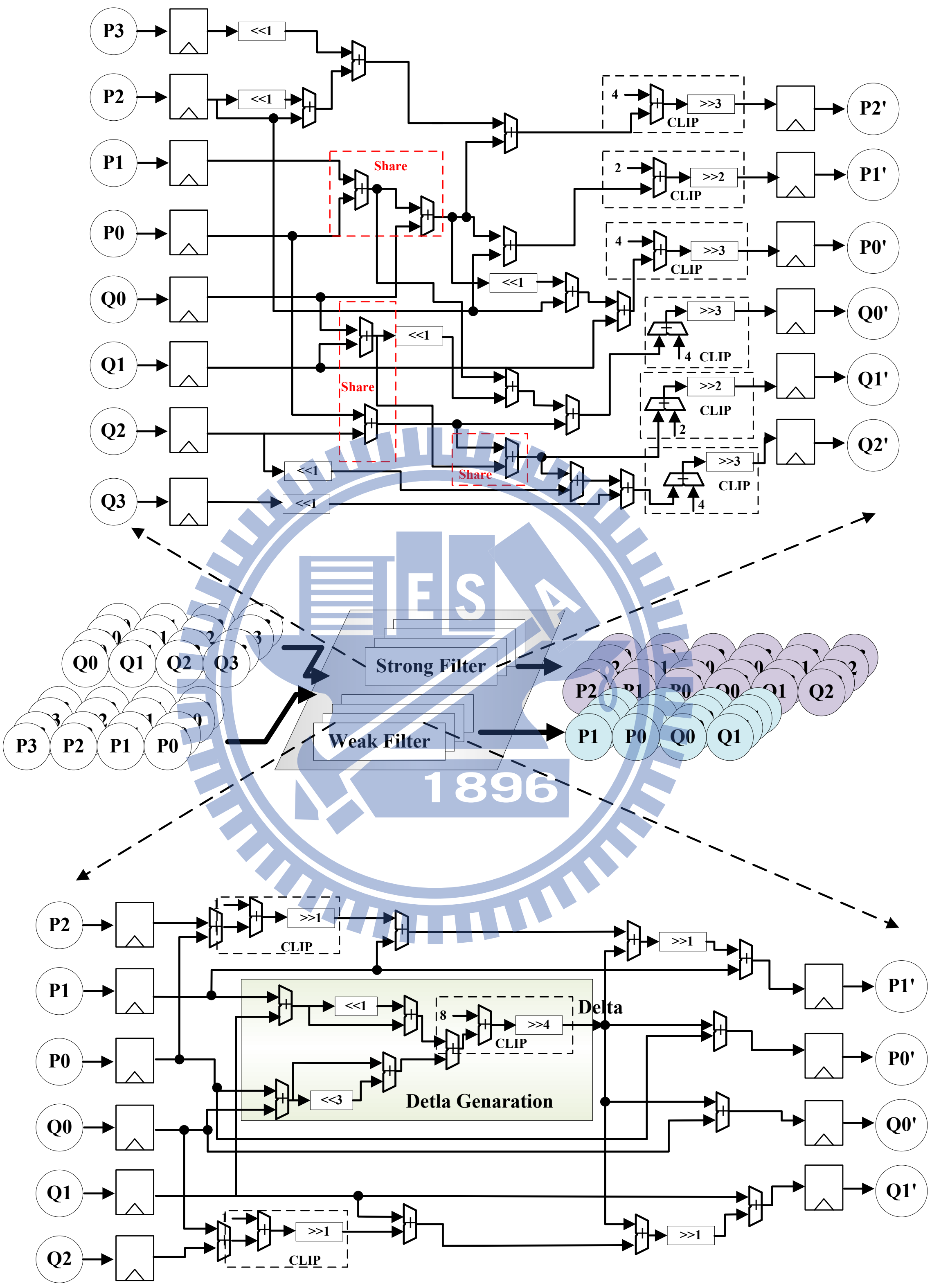


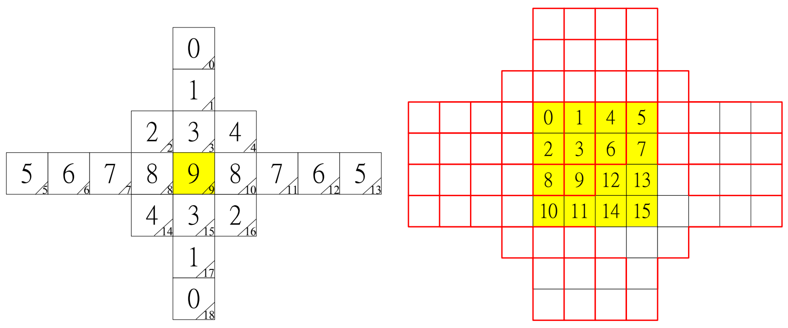
Figure 4.23: Strong and weak filter

4.5 Adaptive Loop Filter

Adaptive loop filter is the novel coding tools in the HEVC which does not exist in H.264. ALF not only further improves video quality but also the bit rate reduction. However, the computational complexity is very huge in the encoder and decoder due to the repeated internal memory accesses. Therefore, in the proposed ALF architecture, the memory accesses need to be reduced further in spite of the long cycle latency.

4.5.1 Merge Filter

In the original adaptive filter type shown in Figure 4.24(a), the 1-pixel output needs 19 number of memory accesses. The repeated memory access times would cost huge power consumption and also would waste cycle count to become a system throughput bottleneck. As a result, if the 16x16 macroblock finished filtering, it needs at most 19×256 memory accesses at the worst case. Therefore, in the proposed filter type, the concept is to make use of the repeated pixels. To reduce the memory accesses times, we adopt 16-pixel filter which needs 76 pixels accesses each 4x4 block. The total memory accesses after finishing the 16x16 macroblock are 76×16 which reduces to 75% totally. Figure 4.24(b) is the 1-pixel filter type merge to the 4x4 filter type. The merger filter not only reduces memory accesses times but also reduce the cycle count for the pipeline limitation.



(a) Adaptive loop filter type (b) Merge the filter to the 4x4 block

Figure 4.24: Adaptive Filter Merge to the 4x4 Block

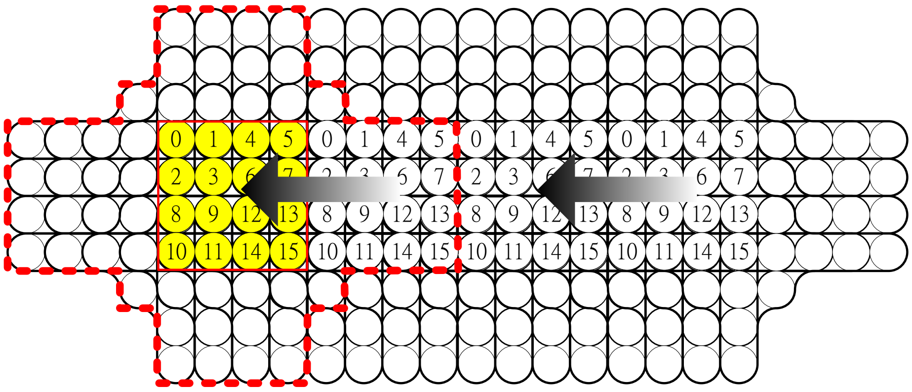
4.5.2 Architecture of the ALF

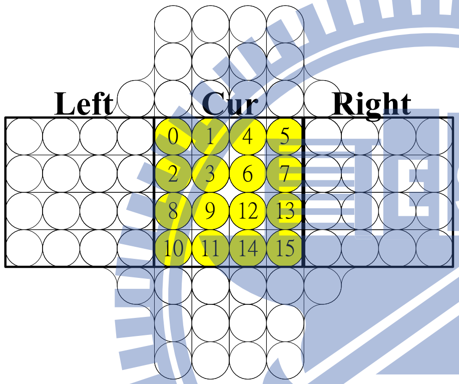
In the proposed architecture of the alf, we first describe the block ordering. We adopt raster scan order rather than double z scan as shown in Figure 4.25(a). The reasons why we did not follow the z scan order is that following the raster scan, the internal memory accesses would be reduced due to reusing the pixels. As shown in Figure 4.25(b), the intermediate pixels from the internal SRAM would be stored into the registers for the temporary used. We depart three types of registers, the left, current, and right registers. During the filtering operation, the pixels in the right registers would be shifted to the current register and also the current would move to the left respectively as shown in Figure 4.25(c). In the interpolation filter in alf, the equation 4.2 is shown that the coefficients are already reconstructed by the entropy decoder. From Figure 4.24(a), the right-bottom number means the pixel index, and the middle number means the coefficient number.

$$\begin{aligned} ALF_{Filtered} = {} & coef0 \times (I_0 + I_{18}) + coef1 \times (I_1 + I_{17}) \\ & + coef2 \times (I_2 + I_{16}) + coef3 \times (I_3 + I_{15}) \\ & + coef4 \times (I_4 + I_{14}) + coef5 \times (I_5 + I_{13}) \\ & + coef6 \times (I_6 + I_{12}) + coef7 \times (I_7 + I_{11}) \\ & + coef8 \times (I_8 + I_{10}) + coef9 \times (I_9) \end{aligned} \tag{4.2}$$

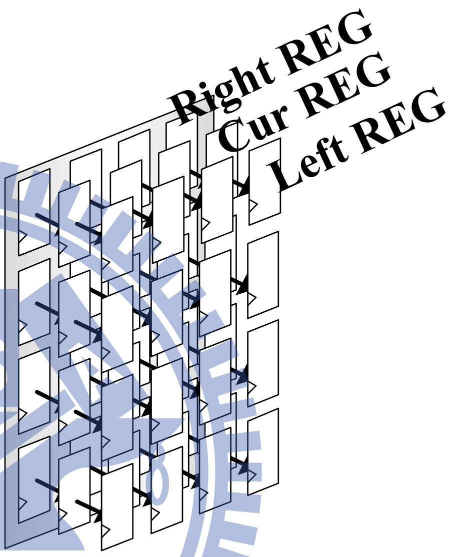
4.6 System Design of the I-Frame Decoder with Wavefront Parallel Processing

In the system design of the I-Frame decoder with WPP, the Figure 4.26 is shown that the inverse transform coder is adopted 1 set, while the intra predictor, deblocking filter and adaptive loop filter are adopted 4 sets to enhance the video throughput for super-high vision $8K \times 4K$. The section below would describe the pipeline stage how to write and the data scheduling of

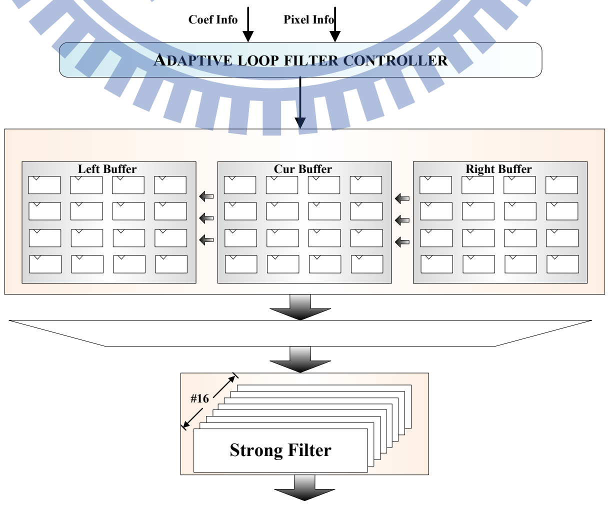

(a) Adaptive loop filter in 16x16 process



(b) Registers to the adaptive loop filter



(c) Registers shifting to the adaptive loop filter



(d) Registers shifting to the adaptive loop filter

Figure 4.25: Adaptive Filter Merge to the 4x4 Block

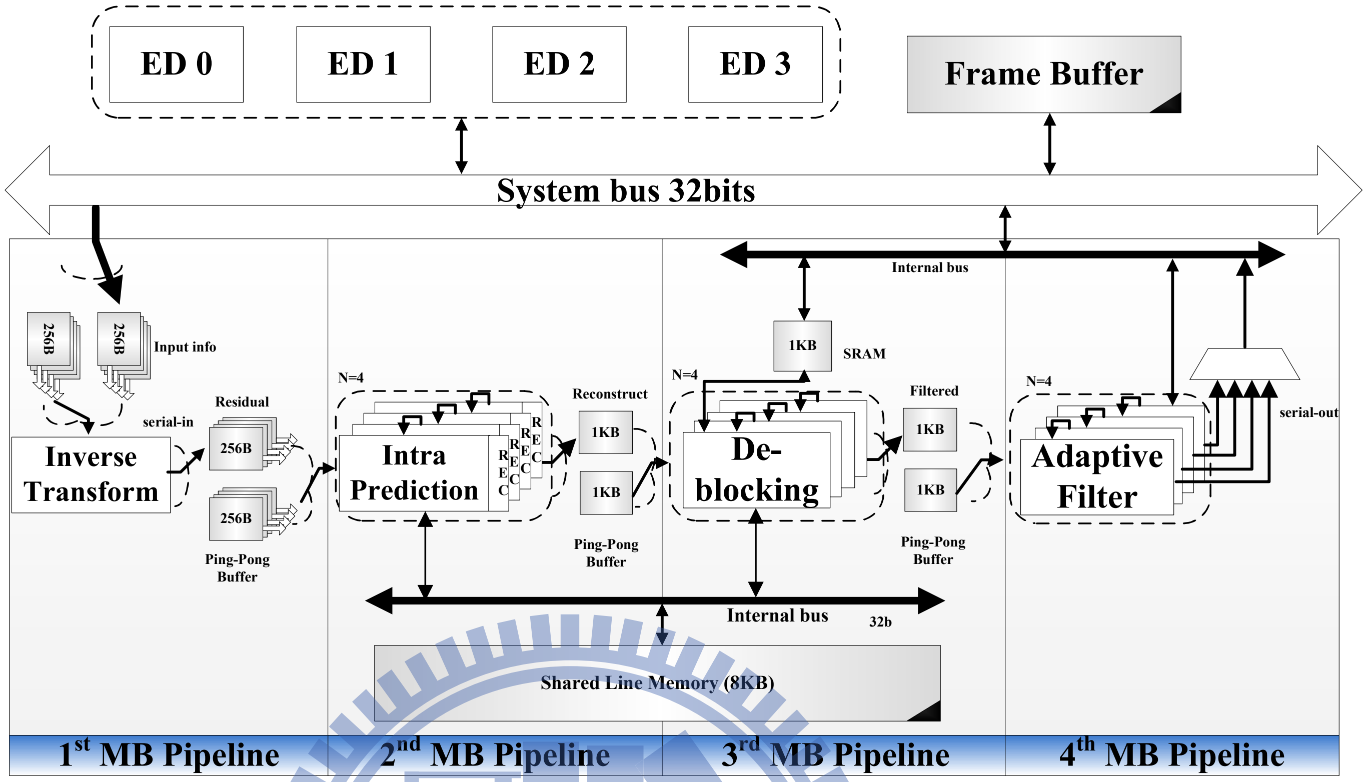


Figure 4.26: System architecture of the WPP

how to use 1-line buffer for intra predictor instead of 4 sets 1-line buffers.

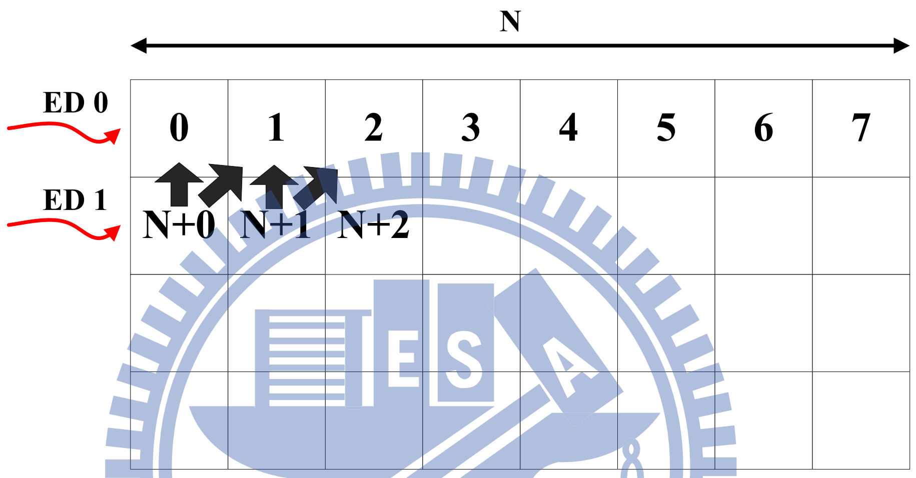
4.6.1 Ping-Pong Buffer

The pipeline would synchronously write the intermediate output to the SRAM which is 256-byte large. With the support of the ping-pong buffer writing and reading, the cycle count could be efficiently used in order to meet the pipeline stage limitation. Each pipeline stage occupies double 256-byte SRAMs and due to the 4-parallel processing, the totally amount of the pipeline buffers are 40-KBytes.

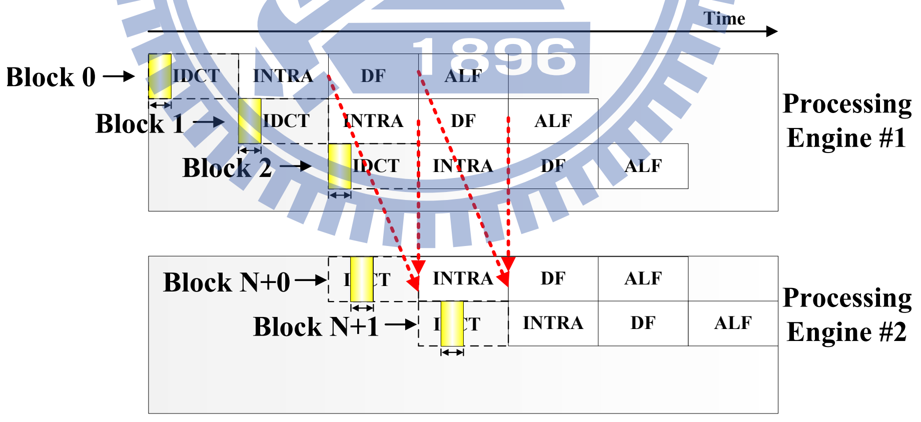
4.6.2 Data Scheduling

Due to the wavefront parallel processing technique in the HEVC, the 4-parallelism of the intra predictor does not requie 4 sets of the 1-line buffer. As shown in Figure 4.27(a), as the top-most macroblock number 2 is started, the next line macroblock $N + 0$ is going to start.

Therefore, the arrow which means the data dependency could still be maintained to meet the HEVC specification. In the proposed hardware, we apply 4-parallel architecture to enhance the throughput. In the memory of the intra predictor, because of the data dependency is still maintained, the reconstruced pixels for the first intra predictor did not need to store into the internal line buffer. Moreover, in Figure 4.27(b), the intra predictor and deblocking filter, adaptive loop filter, all of them can share the data for the next line. Therefore, the memory is applied 1 for the intra predictor.



(a) Wavefront parallel processing



(b) Data scheduling of the WPP

Figure 4.27: Data scheduling of the WPP

4.7 Summary

In conclusion, the I-Frame decoder with 4-parallelism is proposed to achieve $4x$ throughput for meeting the super-high vision $8K \times 4K$. The next chapter would talk about the implementation results of the proposed architecture and the verification platform would also be described.

Chapter 5

Implementation Results and Performance Evaluation

In this chapter, the hardware implementation results would be displayed the chip layout, and the performance including power consumption contribution would also be displayed. We have 3 sections, one is implementation results, another is platform introduction, the other is performance comparison.

5.1 Implementation Results

In the hardware of the implementation, the circuit is used in hardware description language, verilog. Moreover, the final implemention is synthesized using UMC-90nm 1P9M. Our I-Framed decoder can run at highest frequency 320MHz with each stage 350 cycles.

Table 5.1: Chip details

Configuration	Specification
Technology	UMC90nm
Standard	H.265
Throughput	7680x4320 @30fps
Gate Count	832K
Max Frequency	320MHz
Internal SRAM	2.6KB

5.2 Verification Platform

The hardware and software integration would help the hardware verification in real case. As the hardware component is synthesized using Xilinx ISE with place and route, the bit file would finally upload into the Field Programmable Gate Array (FPGA). Besides, the software part is utilizing the high performance ARM CPU to deal with handling efficiently. When the hardware and software is successfully cowork to verify the hardware resource, the output of the results would be displayed on the LCD panel.

5.2.1 Platform Introduction

For integrating the hardware and software resource to co-verify the proposed hardware, we apply the CDK development platform to support the requirement. As shown in Figure 5.1, the CDK board composed of the high performance CPU which is $ARM926EJ-S$ processor running at 266MHz. Moreover, the hardware test chip is also provided by the CDK, the Spartan-3 XC3S4000 could provide 430K gates. Besides, the bus connecting the hardware component and the ARM processor is important for the integration. Therefore, the CDK offers the AMBA 2.0 protocol utlizing Advanced High performance Bus (AHB) to communicate with slave and master part as shown in Table 5.2.

Table 5.2: Hardware CDK peripherals

Item	Peripherals
CPU	AMR926EJ-S
SDRAM	64M-Byte
TFT-LCD	320x240 (RGB)
AHB Freq	50M Hz
FPGA	xc3s4000
Protocol	AMBA 2.0

5.2.2 Demo Results

1. Input Transfering

In the input transfering of the video decoder, the CPU processor is the main character to

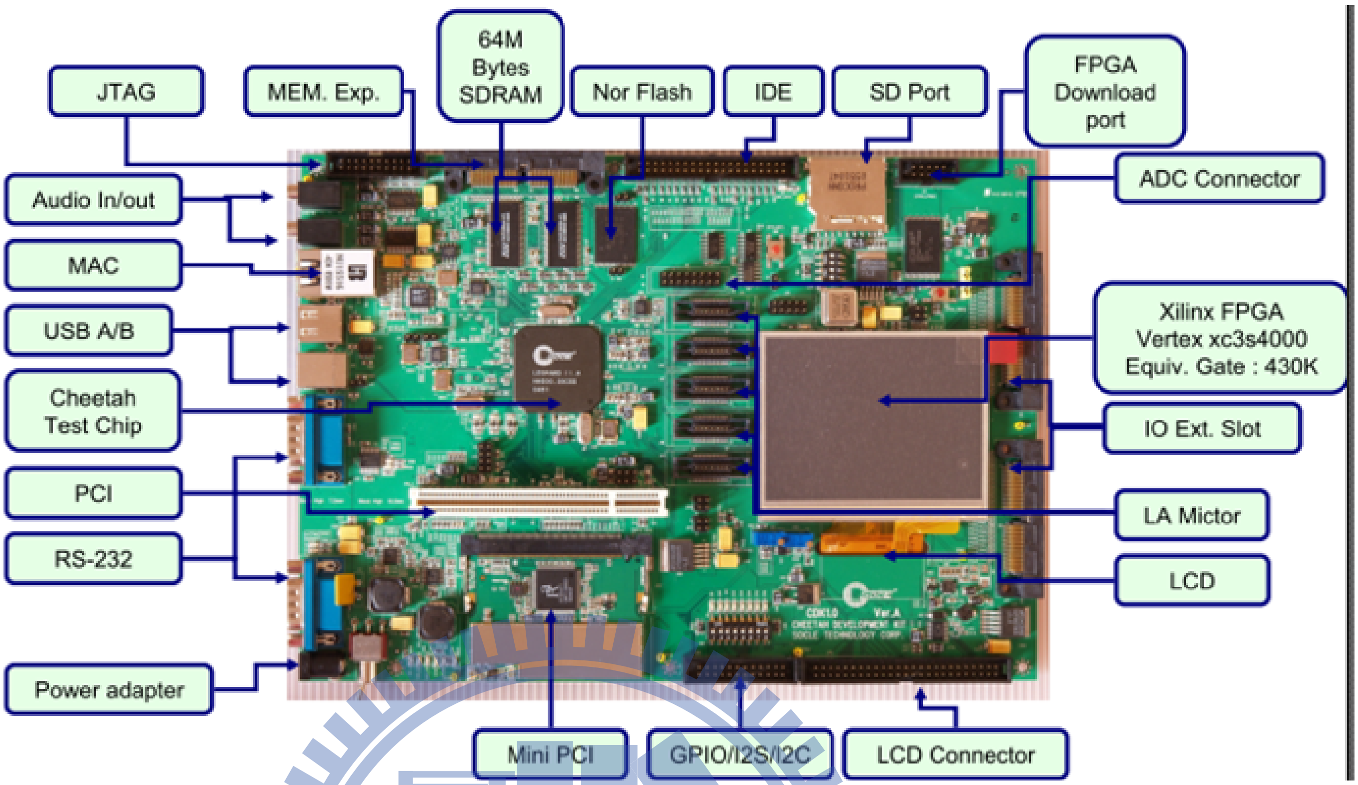


Figure 5.1: Platform overview

handle it. In the software script, we use C programming which is highly portable to the target machine, CDK. Also, the SDRAM controller and LCD panel display controller are also controlled by the CPU processor. The system bus connecting with FPGA which is composed of the hardware gate count is following the AMBA 2.0 protocol.

2. Video Decoder

In the demo architecture,the purpose of the CPU is to process all input controll information and deal with the output results from the FPGA by the AHB. The external memory would store the predicted frames and the filtered frames to prepare for displaying the video sequences. In the hardware part, the AHB slave interface is to receive the input information from the external memory. Moreover, the I-Frame video decoder system is applied 1 set due to the limited FPGA gate counts. As the video decoder pipeline uses 16x16 macroblock, the output results would be stored into the external SDRAM using the connecting AHB. Due to the YUV $4:2:0$, the final results need to be changed to the

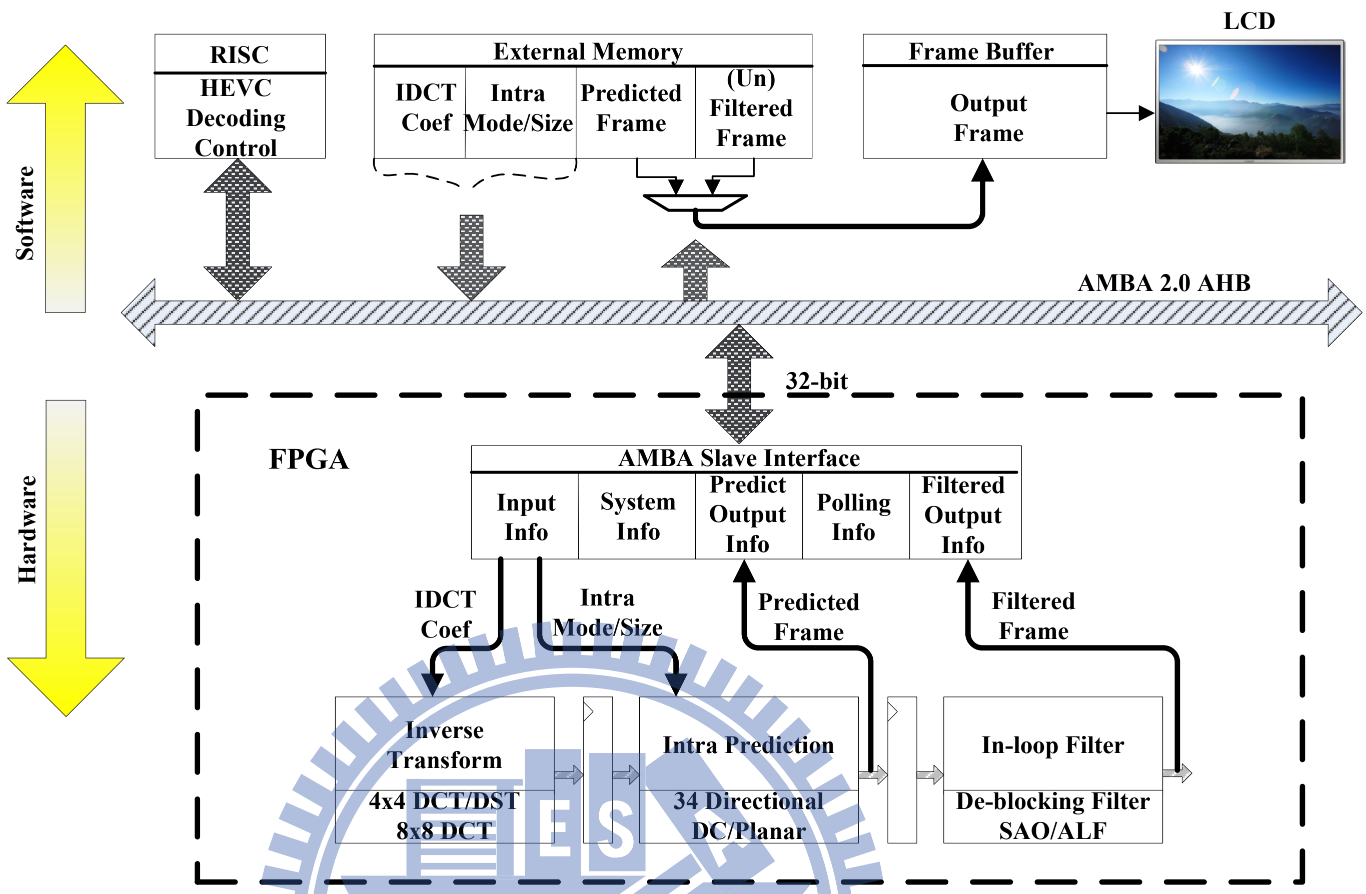


Figure 5.2: Demo architecture

RGB(32bits) type in order for the TFT-LCD to display.

3. Display LCD

After the final results are the RGB(32bits), the CPU processor would controll the the results from accessing external SDRAM to the framebuffer.

4. Design Flow

In Figure 5.3, the proposed demo design flow is shown here. In the beginning, we depart hardware and software type for the function spliting from the HM reference software model. After the hardware modules are verified by the testbench, the verilog code would finally synthesize and place/route by using Xilinx. As the software and hardware are all ready, the bitstream file would be loaded into the FPGA and the software excutable file would check whether the LCD results right or not.

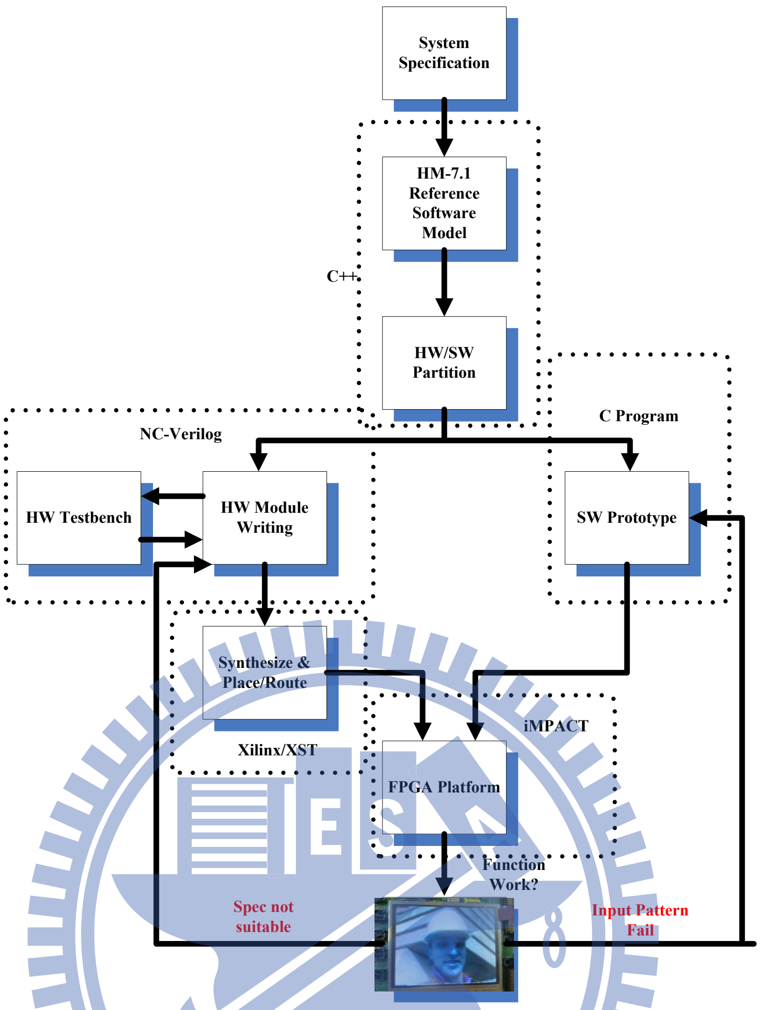


Figure 5.3: Demo design flow

5. **Demo Pattern**

   In the pattern case, we test 3 patterns which are foreman, carphone, and mother/daughter. As shown in Figure5.4, the demo platform is shown that our excutable file is transfered by the USB. The data in the terminal are coming from the RS-232. Figure 5.5 is shown the final LCD results for the intra predicted and loop filtered.

5.3 Performance Comparison

This section of the performance comparison contains power consumption, miss rate analysis comparison and the hardware comparison which are all compared with the state-of-the-arts [27] and [2].

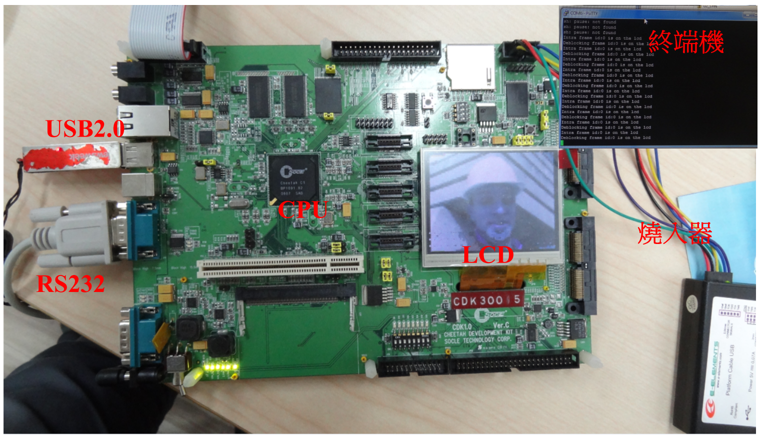


Figure 5.4: Demo platform

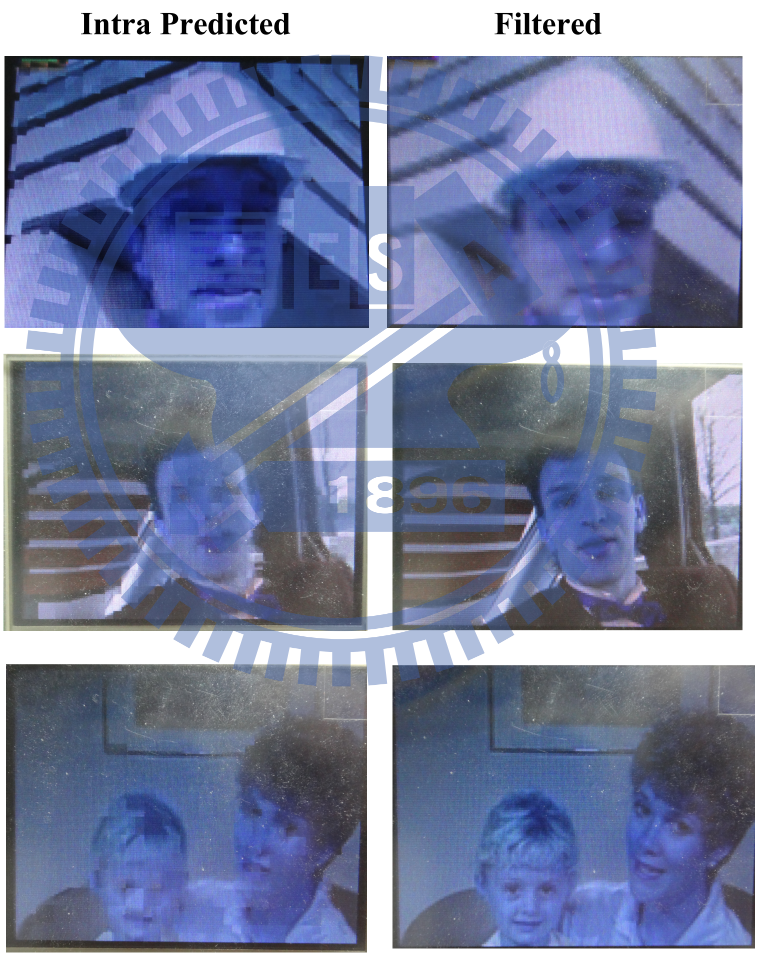


Figure 5.5: Demo results

5.3.1 Power Consumption

In our first tape-out of the HEVC video decoder, the power profiling of the video decoder components are shown here in Figure 5.6. The internal memory used in the HEVC decoder consumes about 36% of all in 97.4mW running at 294MHz, and in the memory power of 83% occupies the deblocking filter and intra predictor memory power consumption. Therefore, if

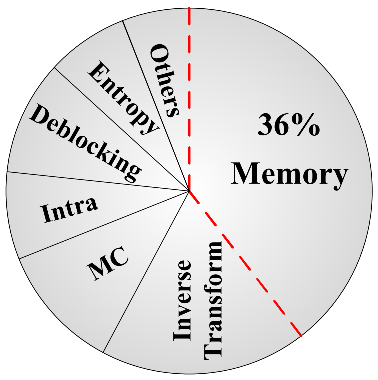


Figure 5.6: Power profile of the HEVC decoder

we apply our proposed method including shared above line buffer and exploit prediction-based memory hierarchy to the completely HEVC video decoder, the total core power reduction could achieve about 19.4% of all with 60% memory reduction for the full-HD resolution as shown in Figure 5.7. For the fixed memory reduction factor set as 8, the miss penalty for the full-HD is

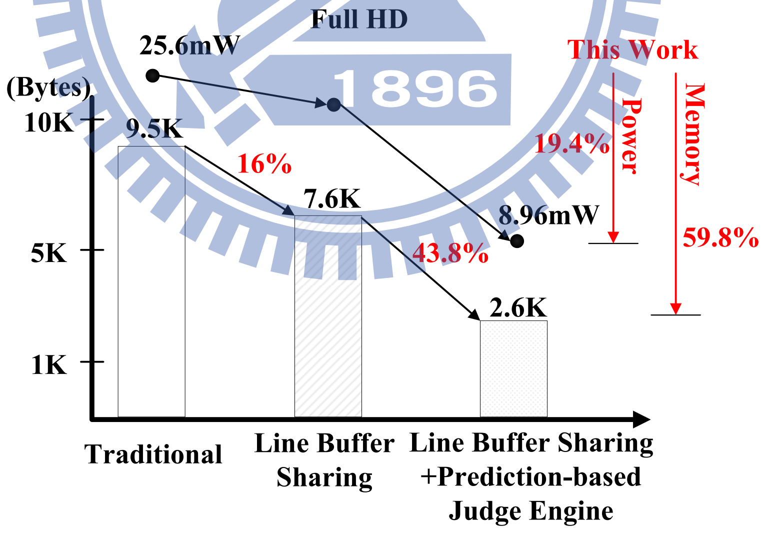


Figure 5.7: Memory and power distribution of the proposed algorithm

shown in next section.

5.3.2 Hit Rate Comparison

In the hit rate comparison with [1], the sequence name are list below with the memory reduction factor 2 to 8. In the Figure 5.8, the hit rate could be improved from the 57.2% to 6.6% range. Further, in the external memory bandwidth comparison, Figure 5.9 shows that the the

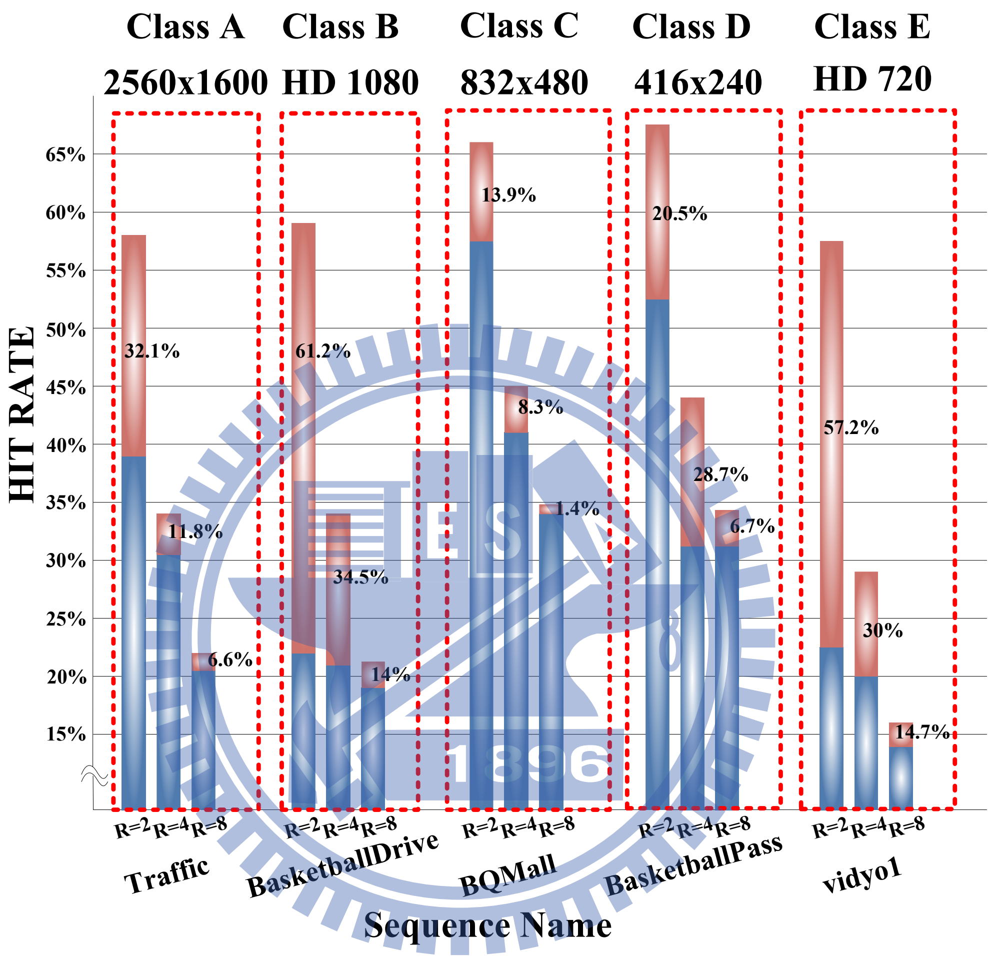


Figure 5.8: Hit rate analysis comparison

reduction ratio of the external memory bandwidth is from 70.2% to the 32%.

5.3.3 Hardware Comparison

In the hardware comparison Table 5.3 shown, we list 2 state-of-the-arts compared our proposed work. In the [27], the memory reduction factor is set 8 and its hit rate could achieve 19.2% while our work could achieve better than his design for the 22.4%. Therefore, the miss penalty of the external memory bandwidth access is less than 9.5MB/sec for the 6.09MB/sec.

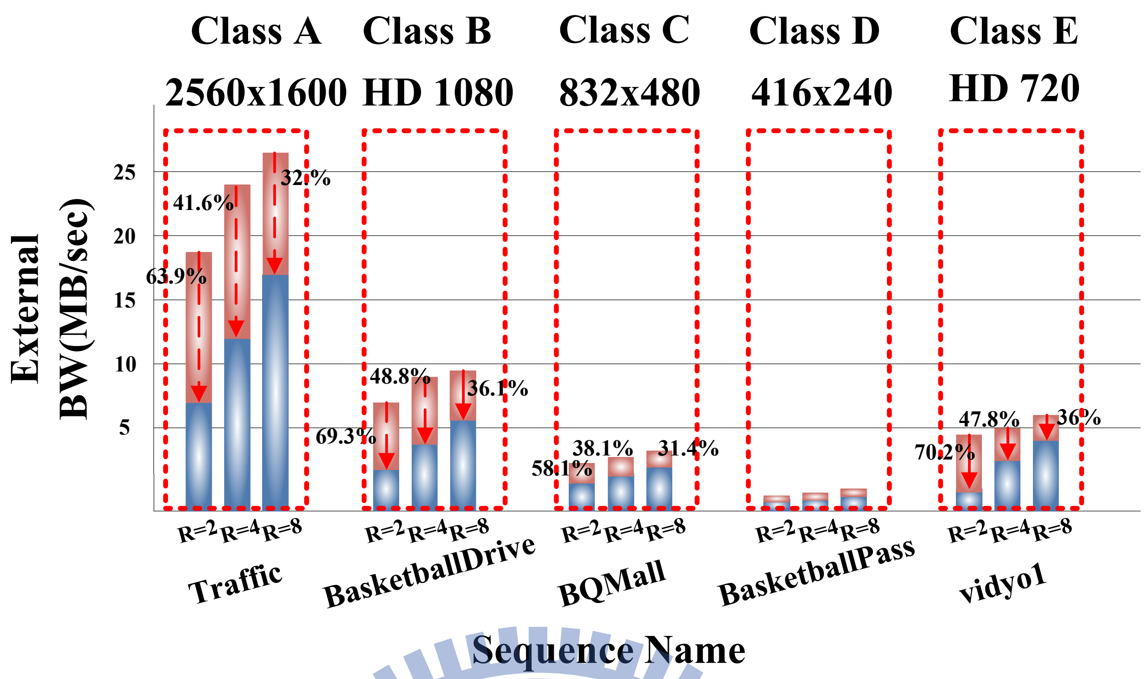


Figure 5.9: External bandwith comparison

In [2], although the SRAM is the least of all designs, but external memory bandwidth is the largest and also the SDRAM power consumption.

5.4 Summary

In the summary of the implementation results, the proposed work could reduce internal memory of the deblocking filter and intra predictor for about 85% reduction ratio in the Full-HD resolution and the core power could be reduced to the 50% with only 3.6mW SDRAM power in the miss penalty. Also, the proposed hardware design could process $8K \times 4K$ resolution with high throughput by utilizing the wavefront parallel processing.

Table 5.3: Hardware comparison

	ISSCC '07 [27]	ISSCC '07 [2]	Proposed
Target Video Resolution	1920x1080 30fps	1920x1080 30fps	7680x4320 30fps
Max Frequency	100MHz	120MHz	320MHz
Normalized Throughput @100MHz	62.2M pixels/sec	51.8M pixels/sec	312.5M pixels/sec
Gate Count (Intra+In-loop +Idct)	400K	115K	832K
Algorithm	Line-Pixel Lookahead	DMA Buffer	Share Line Buffer +Prediction-based
Memory Reduction	$(5\times Width)$ $\times 12.5\%$ (Factor 8)	N/A	1-line buffer $(3\times Width)$ $\times 12.5\%$(Factor 8)
SRAM (Intra+In-loop)	1.2KB	0.38KB	2.6KB
Hit Rate	19.2%	N/A	22.4%
External BW (Miss Penalty)	9.5MB/sec	83.4MB/sec	6.09MB/sec
SDRAM Power (Miss Penalty)	5.7mW	33.9mW	3.6mW

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we implemented an I-Frame decoder with parallelism technique and memory reduction method to achieve low memory, high throughput HEVC video decoder.

From the system point of view, first considering the memory requirement in the total video decoder, the intra predictor and deblocking filter occupy almost 83% totally and the power consumption occupy about 15.3% of all. Therefore, with higher resolution, the memory requirements and power consumption would harm the hardware processing. By the proposed shared above line buffer for the deleting 1-line buffer for deblocking filter, it could save the internal memory to the 20% reduction ratio of all. Further, the prediction-based memory hierarchy for containing $1/8$ frame size of internal SRAM for the deblocking filter is limited for the system point considering the external memory bandwidth and SDRAM power consumption. Compared with the [27], the hit rate could achieve better performance in the SRAM reduction factor set as 8. In Figure 5.8, the hit rate is improved about 14% in the Full-HD sequence. While the external memory bandwidth in the miss penalty is reduced about the 36.1% in the Full-HD sequence. For accessing the external memory, the power consumes in the SDRAM would also be considered to the system. In Table 5.3, the SDRAM power could also be reduced 36% for only 3.6mW. In the proposed architecture, the I-frame decoder consists of the transform coder, intra predictor and in-loop filter which are improved the throughput which can be enhanced to decode 8Kx4K super-high vision by using the wavefront parallel processing and the internal memories could be reduced using the shared above line buffer and the second level memory hierarchy, as a result, the proposed work could be applied to the super-high vision mulimedia application and the low power portable devices.

6.2 Future Work

In the future work, the application of the multimedia video codec could be applied to the intelligent video surveillance (IVS). The engine camera could track and record the motion of the objects and also identify the information of the objects. Moreover, as the information of the video could appeal to the viewers, the decoder must combine with the detection engine used to specify the human's gestures and also the cloud computing is the important technique to complete this application.

Bibliography

[1] T.-M. Liu and C.-Y. Lee, "Memory-hierarchy-based power reduction for h. 264/avc video decoder," in *VLSI Design, Automation and Test, 2006 International Symposium on*, 2006, pp. 1–4.

[2] C.-C. Lin, J.-W. Chen, H.-C. Chang, Y.-C. Yang, Y.-H. O. Yang, M.-C. Tsai, J.-I. Guo, and J.-S. Wang, "A 160k gates/4.5 kb sram h.264 video decoder for hdtv applications," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 1, pp. 170–182, 2007.

[3] J. Shah, K. S. Raghunandan, and K. Varghese, "Hd resolution intra prediction architecture for h.264 decoder," in *VLSI Design (VLSID), 2012 25th International Conference on*, 2012, pp. 107–112.

[4] D. Zhou, G. He, W. Fei, Z. Chen, J. Zhou, and S. Goto, "A 4320p 60fps h.264/avc intra-frame encoder chip with 1.41gbins/s cabac," in *VLSI Circuits (VLSIC), 2012 Symposium on*, 2012, pp. 154–155.

[5] Y.-C. Chao, J.-K. Lin, J.-F. Yang, and B.-D. Liu, "A high throughput and data reuse architecture for h.264/avc deblocking filter," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, 2006, pp. 1260–1263.

[6] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C. Wang, T.-H. Chang, and L.-G. Chen, "Architecture design for deblocking filter in h.264/jvt/avc," in *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, vol. 1, 2003, pp. I–693–6 vol.1.

[7] S.-C. Chang, W.-H. Peng, S.-H. Wang, and T. Chiang, "A platform based bus-interleaved architecture for de-blocking filter in h.264/mpeg-4 avc," *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 1, pp. 249–255, 2005.

[8] B. Sheng, W. Gao, and D. Wu, "An implemented architecture of deblocking filter for h.264/avc," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 1, 2004, pp. 665–668 Vol. 1.

[9] C.-C. Cheng, T.-S. Chang, and K.-B. Lee, "An in-place architecture for the deblocking filter in h.264/avc," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 7, pp. 530–534, 2006.

[10] M. Sima, Y. Zhou, and W. Zhang, "An efficient architecture for adaptive deblocking filter of h.264/avc video coding," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 1, pp. 292–296, 2004.

[11] T.-W. Chen, Y.-W. Huang, T.-C. Chen, Y.-H. Chen, C.-Y. Tsai, and L.-G. Chen, "Architecture design of h.264/avc decoder with hybrid task pipelining for high definition videos,"

in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 2931–2934 Vol. 3.

[12] D. Zhou, J. Zhou, X. He, J. Zhu, J. Kong, P. Liu, and S. Goto, "A 530 mpixels/s 4096x2160@60fps h.264/avc high profile video decoder chip," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 4, pp. 777–788, 2011.

[13] D. Zhou, J. Zhou, J. Zhu, P. Liu, and S. Goto, "A 2gpixel/s h.264/avc hp/mvc video decoder chip for super hi-vision and 3dtv/ftv applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, 2012, pp. 224–226.

[14] T.-A. Lin, S.-Z. Wang, T.-M. Liu, and C.-Y. Lee, "An h.264/avc decoder with 4 times;4-block level pipeline," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 1810–1813 Vol. 2.

[15] D. Zhou, Z. You, J. Zhu, J. Kong, Y. Hong, X. Chen, X. He, C. Xu, H. Zhang, J. Zhou, N. Deng, P. Liu, and S. Goto, "A 1080p@60fps multi-standard video decoder chip designed for power and cost efficiency in a system perspective," in *VLSI Circuits, 2009 Symposium on*, 2009, pp. 262–263.

[16] T.-D. Chuang, P.-K. Tsung, P.-C. Lin, L.-M. Chang, T.-C. Ma, Y.-H. Chen, and L.-G. Chen, "A 59.5mw scalable/multi-view video decoder chip for quad/3d full hdtv and video streaming applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, 2010, pp. 330–331.

[17] C.-T. Huang, M. Tikekar, C. Juvekar, V. Sze, and A. Chandrakasan, "A 249mpixel/s hevc video-decoder chip for quad full hd applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, 2013, pp. 162–163.

[18] K.-H. Chen, J.-I. Guo, K.-C. Chao, J.-S. Wang, and Y.-S. Chu, "A high-performance low power direct 2-d transform coding ip design for mpeg-4 avc/h.264 with a switching power suppression technique," in *VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on*, 2005, pp. 291–294.

[19] C.-T. Lin, Y.-C. Yu, and L.-D. Van, "Cost-effective triple-mode reconfigurable pipeline fft/ifft/2-d dct processor," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 8, pp. 1058–1071, 2008.

[20] C.-C. Sun, P. Donner, and J. Gotze, "Low-complexity multi-purpose ip core for quantized discrete cosine and integer transform," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2009, pp. 3014–3017.

[21] Y.-F. Lai and Y.-K. Lai, "Design and implementation of reconfigurable idct architecture for multi-standard video decoders," in *SoC Design Conference (ISOCC), 2010 International*, 2010, pp. 107–110.

[22] Y.-H. Chen and T.-Y. Chang, "A high performance video transform engine by using space-time scheduling strategy," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 4, pp. 655–664, 2012.

[23] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/avc baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 704–716, 2003.

[24] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, “Design of an image edge detection filter using the sobel operator,” *Solid-State Circuits, IEEE Journal of*, vol. 23, no. 2, pp. 358–367, 1988.

[25] T.-M. Liu, W.-P. Lee, T.-A. Lin, and C.-Y. Lee, “A memory-efficient deblocking filter for h.264/avc video coding,” in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 2140–2143 Vol. 3.

[26] M. Nadeem, S. Wong, G. Kuzmanov, A. Shabbir, M. Nadeem, and F. Anjam, “Low-power, high-throughput deblocking filter for h.264/avc,” in *System on Chip (SoC), 2010 International Symposium on*, 2010, pp. 93–98.

[27] T.-M. Liu, T.-A. Lin, S.-Z. Wang, W.-P. Lee, J.-Y. Yang, K.-C. Hou, and C.-Y. Lee, “A 125 956;w , fully scalable mpeg-2 and h.264/avc video decoder for mobile applications,” *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 1, pp. 161–169, 2007.