

電子工程學系 電子研究所碩士班

碩 士 論 文

針對 H.264/AVC 去方塊濾波器及框內編碼之  
演算法和架構設計

Algorithm and Architecture Design for H.264/AVC  
Deblocking Filter and Intra Coding



研究生： 鄭朝鐘

指導教授： 張添烜 博士

中華民國 九十四 年 六 月

針對 H. 264/AVC 去方塊濾波器及框內編碼之  
演算法和架構設計

Algorithm and Architecture Design for H.264/AVC  
Deblocking Filter and Intra Coding

研究生：鄭朝鐘  
指導教授：張添烜 博士

Student: Chao-Chung Cheng  
Advisor: Dr. Tian-Sheuan Chang

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 Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

in

Electronics Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中華民國 九十四 年 六 月

# 針對 H.264/AVC 去方塊濾波器及框內編碼之 演算法和架構設計

研究生：鄭朝鐘

指導教授：張添烜 博士

國立交通大學

電子工程學系 電子研究所碩士班

## 摘要

數位視訊科技已在我們的日常生活中扮演重要的角色，編碼效能也隨著技術的演進而提升，H.264/AVC 是目前最新的國際視訊編碼標準，相較於 MPEG-4、H.263、和 MPEG-2，分別可節省 39%、49%、和 64% 的資料量，但由於其具有相當複雜之編碼技術及模式選擇，使得運算複雜度也遠高於先前之編碼標準，因此如何設計高效能的運算模組與在不致犧牲 H.264/AVC 之編碼效能之前提下，降低其運算複雜度，為目前相當重要之課題。本論文中，我們的貢獻主要有三個部分，分別是針對 H.264/AVC 系統中：方塊濾波器的架構設計、快速框內預測演算法、以及框內編碼器之架構設計。

去方塊濾波器是 H.264/AVC 視訊編碼系統中的重要模組，用來減少方塊視覺效應，以增進影像品質。佔有不可忽視的運算量，本論文中，我們提出了兩種不同硬體架構，藉由妥善安排資料處理的順序，在不影響輸出結果的情況下，達到更有效的資料利用率與加速處理的效能，和之前的設計相比，第一種架構有控制邏輯簡單的優點，大量的減少控制電路的邏輯閘數目，並減少 50% 的內部記憶體，第二種架構則可以減少 90% 的內部記憶體，並達到更快的運算效率。

框內預測利用空間中資料數值的相關性，用來預測將被編碼的資料數值，是 H.264/AVC 視訊編碼系統中框內編碼的重要利器，在本論文中，我們針對 H.264/AVC 框內預測提出一個簡單的三步驟演算法，利用各預測模式的方向關係，省略出現機率較低之模式的運算，而整個過程，只固定需要運算六個模式，而不像全域搜尋演算法需要找九種模式。和全域搜尋法相比，約可節省約 33% 的框內預測運算量，而只損失約 1% 左右的位元率。

最後，我們提出 H.264/AVC 框內編碼器的硬體演算法及其架構，所提出的硬體演算法省去複雜的平面預測模式，減少佔整體面積最大的框內預測模組，且藉由改善的代價函數來增進壓縮的效能。配合高效能的硬體架構和運算流程，可以在 117.28MHz 下，進行即時的 HDTV(1280x720) 30fps 編碼。

簡而言之，我們對 H.264/AVC 視訊編解碼系統的貢獻主要有三個部分。我們提出的去方塊濾波器架構可以更有效地加速去方塊處理；快速框內預測演算法可以有效減少預測所需的運算量；我們所提出的框內編碼架構可以加快框內編碼的速度。

# Algorithm and Architecture Design for H.264/AVC Deblocking Filter and Intra Coding

Student: Chao-Chung Cheng

Advisor: Dr. Tian-Sheuan Chang

Department of Electronics Engineering & Institute of Electronics

National Chiao Tung University

## Abstract

Digital video technology has played an important role in our daily life. With the evolution of video technology coding efficiency has been greatly improved. H.264/AVC is the latest international video coding standard that can save 39%, 49%, and 64% of bit-rates in comparison with MPEG-4, H.263, and MPEG-2, respectively. However, this efficiency comes with the cost of much higher computational complexity than previous standards due to the complex coding approaches and mode decision techniques. Thus, how to design high performance functional units and reduce computational complexity without too much degradation in coding efficiency are very important topics. In this thesis, we have three contributions for the H.264/AVC design, architecture design of the deblocking filter, a fast intra prediction algorithm, and an architecture design of intra coding in H.264/AVC.

Deblocking filter is an important component of H.264/AVC to reduce the blocking effect and to improve the video quality. It is both computational and memory extensive. In this thesis, two different architecture of deblocking filter are proposed. The computing flow is reordered for efficient data reusability and high throughput while maintain standard compatibility. In the first version, gate count is greatly reduced by simple control unit, and internal memory is also reduced to 50% of that in the previous design. In the second version, the proposed architecture can reduce 90% of internal memory and achieve higher throughput than others.

Intra prediction, which uses the information of spatial correlation to predict the data to be encoded, is an important tool of intra frame coding. In this thesis, we propose a simple fast three step algorithm. The algorithm uses the directional relationship of prediction modes to skip the modes with less probability. Thus, the proposed algorithm can complete the 4x4 intra prediction by only examining six modes instead of nine modes in the full search algorithm. The simulation result shows that the proposed algorithm can maintain similar PSNR quality to that in the full search algorithm with 33% of computation reduction of intra prediction process and only 1% of bit-rate increase.

Finally, a hardware oriented algorithm of intra coding and its architecture are proposed. We save the complex and hardware costly plane mode, which occupies the biggest area in the intra prediction unit in the intra coding and improve the coding efficiency with the enhanced cost function. With well designed high performance functional unit and computing schedule, the proposed architecture can easily support real-time intra coding of HDTV 1280x720@30fps video application when clocked at 117.28MHz.

In brief, our contribution to H.264/AVC video coding system is in three parts. The first contribution to the deblocking filter architecture can accelerate the deblocking process. The second contribution to the fast intra coding algorithm can reduce the computational complexity of intra prediction. The final contribution to the intra coding architecture can speed up the computation of intra frame coding.

## 誌謝

首先，要感謝我的指導教授—張添烜博士，這一年多來給我支持和鼓勵，讓我在研究上能自由的發揮，每當我有困難或疑問時，總會抽空和我詳談，以鼓勵的態度支持我的想法，感激之情，非短短文句可以表達。

也要謝謝我的口試委員們，工研院任建葳主任，交大電子李鎮宜系主任，交大資工蔡淳仁教授，感謝你們百忙中抽空來指導我，因為你們寶貴的意見讓我的論文更加完備。

接著要感謝 VSP 實驗室的好伙伴們。謝謝引我入門的李坤儉學長，教導不少經驗的張彥中學長、許惠錚學姐與林昕儀學姐，你們傳給我許多的經驗，讓我受用不盡。感謝林佑昆學長、秦浩雲學長與史彥芪同學，參加 SIP 競賽的過程，一起加油打氣，熬夜而建立出的革命情感，也讓我學習了不少的經驗。感謝海珊學長、陳漢臣學長、楊智喬學長、君偉、旻奇、裕仁、國巨、錦木、得瑋學弟們，有你們的幫忙，讓我在實驗室的生活都能順順利利。還有最棒的戰友林亭安，從大四開始，經過不懈的努力，我們終於在 IC 設計競賽獲得了不錯的成績。所有的一切，都是我在交大寶貴的回憶。

最後要感謝默默支持我的家人們，我的爸媽、姐姐、妹妹、你們的溫暖是我努力最大的支柱。

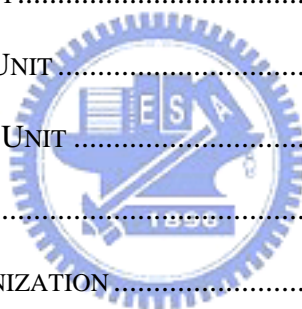
在此，把這本論文獻給所有愛我與所有我愛的人

## Contents

<b>CHAPTER 1 INTRODUCTION .....</b>	<b>1</b>
1.1. MOTIVATION .....	1
1.2. THESIS ORGANIZATION .....	2
<b>CHAPTER 2 OVERVIEW OF H.264/AVC STANDARD.....</b>	<b>3</b>
2.1. INTRODUCTION TO H.264/AVC.....	3
2.2. PROFILE AND LEVEL.....	7
<b>CHAPTER 3 ARCHITECTURE DESIGN OF DEBLOCKING FILTER IN H.264.....</b>	<b>9</b>
3.1. FUNDAMENTAL OF H.264/AVC DEBLOCKING FILTER .....	10
3.2. ARCHITECTURE DESIGN OF H.264/AVC DEBLOCKING FILTER .....	13
3.2.1. VERSION 1 .....	13
3.2.1.1. SIMPLE DATA FLOW.....	13
3.2.1.2. HARDWARE ARCHITECTURE.....	14
3.2.2. VERSION 2 .....	18
3.2.2.1. FULL DATA REUSE FLOW.....	18
3.2.2.2. IN-PLACE DEBLOCKING FILTER ARCHITECTURE.....	21
3.2.2.3. MEMORY ORGANIZATION.....	22
3.2.2.4. PROCESSING SCHEDULE .....	23
3.3. IMPLEMENTATION AND COMPARISON.....	25
3.4. SUMMARY OF PROPOSED ARCHITECTURES.....	26
<b>CHAPTER 4 FAST 4X4 INTRA PREDICTION ALGORITHM FOR H.264/AVC .....</b>	<b>27</b>
4.1. FUNDAMENTAL OF H.264/AVC 4x4 INTRA PREDICTION.....	27
4.2. REVIEW OF PREVIOUS APPROACHES.....	30
4.3. FAST THREE STEP INTRA PREDICTION ALGORITHM.....	31
4.4. SIMULATION RESULT AND DISCUSSION .....	32



4.5.	SUMMARY OF PROPOSED INTRA PREDICTION ALGORITHM .....	37
<b>CHAPTER 5</b>	<b>ARCHITECTURE DESIGN FOR H.264/AVC INTRA CODING .....</b>	<b>38</b>
5.1.	FUNDAMENTAL OF H.264/AVC INTRA CODING .....	39
5.1.1.	INTRA PREDICTION MODE.....	40
5.2.	HARDWARE ORIENTED ALGORITHM MODIFICATION .....	40
5.2.1.	PROPOSED MODE DECISION METHOD.....	40
5.2.2.	INTRA PREDICTION MODE.....	43
5.3.	ARCHITECTURE DESIGN OF H.264/AVC INTRA CODING .....	49
5.3.1.	SYSTEM ARCHITECTURE DESIGN .....	49
5.3.2.	INTRA PREDICTOR GENERATION UNIT .....	51
5.3.3.	TRANSFORM UNIT.....	55
5.3.4.	QUANTIZATION UNIT.....	57
5.3.5.	MODE DECISION UNIT .....	58
5.3.6.	CAVLC UNIT .....	59
5.3.7.	MEMORY ORGANIZATION .....	59
5.3.8.	OVERALL ARCHITECTURE PERFORMANCE .....	60
5.4.	IMPLEMENTATION RESULTS.....	61
<b>CHAPTER 6</b>	<b>CONCLUSION .....</b>	<b>63</b>
<b>BIBLIOGRAPHY</b>	<b>.....</b>	<b>64</b>



## List of Tables

Table 1	Parameters for determining boundary strength.....	12
Table 2.	Comparison of cost synthesized at 100MHz. (excluding memory cost). .....	25
Table 3.	Comparison of memory size.....	26
Table 4.	Comparison of processing capability.....	26
Table 5.	QP = 28, Comparison results.....	33
Table 6.	QP = 32, Comparison results.....	33
Table 7.	QP = 36, Comparison results.....	34
Table 8.	QP = 40, Comparison results.....	34
Table 9.	List of gate count .....	61



## List of Figures

Fig. 1 Block diagram of H.264/AVC encoder .....	4
Fig. 2 Block size of motion estimation/compensation .....	4
Fig. 3 Subjective view comparison of picture with deblocking filter (left) and without deblocking filter (right).....	6
Fig. 4 R-D curve comparison of H.264/AVC with MPEG-4, H.263, and MPEG-2 .....	7
Fig. 5 Profile of H.264/AVC .....	8
Fig. 6 Encoding loop of H.264.....	10
Fig. 7. Original processing flow for (a)horizontal filtering, and (b)vertical filtering .....	12
Fig. 8 Convention for describing samples across two 4x4 block boundary...	12
Fig. 9. Modified processing flow for (a) horizontal filtering, and (b) vertical filtering .....	14
Fig. 10. Data structure of deblocking filter .....	14
Fig. 11. The proposed deblocking filter architecture. ....	15
Fig. 12. 4x4 Block index for one macroblock.....	15
Fig. 13 Data path for (a) horizontal filtering and (b) vertical filtering.....	16
Fig. 14 timing diagram of simple data flow deblocking filter .....	17
Fig. 15. Block index .....	18
Fig. 16. Edge processing order for (a) luma edge, and (b) chroma edge .....	19
Fig. 17. Overall architecture.....	19
Fig. 18. Data path (a) horizontal filtering over vertical edges, and (b) vertical filtering over horizontal edges.....	20
Fig. 19 Organization of on-chip 1R/1W port SRAM.....	22
Fig. 20. Data flow of deblocking filter (a) processing the left vertical edges, (b)	

processing vertical edges, and (c) processing horizontal edges.....	24
Fig. 21. Direction of 9 4x4 intra prediction modes in H.264.....	28
Fig. 22. A 4x4 block and its neighboring pixels.....	28
Fig. 23. 9 mode of 4x4 intra prediction .....	28
Fig. 24. Adjacent block of current 4x4 block .....	29
Fig. 25. Flow chart of three step intra algorithm .....	32
Fig. 26. RD-curve of mobile & calendar .....	35
Fig. 27. RD-curve of foreman.....	35
Fig. 28. RD-curve of Stefan.....	36
Fig. 29. RD-cure of news.....	36
Fig. 30. RD-curve of coastguard.....	37
Fig. 31. Flow of H.264/AVC intra coding.....	38
Fig. 32. Modes of Intra4x4.....	39
Fig. 33. Modes of Intra16x16.....	39
Fig. 34. Modes of chroma8x8.....	40
Fig. 35. 4X4 DCT transform of H.264/AVC.....	41
Fig. 36. quant_coef table of quantization .....	42
Fig. 37. dequant_coef table of inverse quantization. ....	42
Fig. 38. Modified SATD calculation method .....	43
Fig. 39 Four types of intra prediction modes.....	44
Fig. 40. Equations of plane mode prediction .....	44
Fig. 41. RD curve of Akiyo .....	45
Fig. 42. RD curve of Foreman .....	45
Fig. 43. RD curve of container .....	46
Fig. 44. RD curve of stefan.....	46
Fig. 45. RD curve of football.....	47

Fig. 46. RD curve of mobile and calendar .....	47
Fig. 47. RD curve of tempete .....	48
Fig. 48. RD curve of news.....	48
Fig. 49 Architecture of Intra Coding .....	49
Fig. 50. Reconfigurable data path of intra predictor generation unit .....	52
Fig. 51. Data path of diagonal down right.....	52
Fig. 52. Data path of vertical right .....	53
Fig. 53. Data path of horizontal down mode.....	53
Fig. 54. Data path of DC prediction mode .....	54
Fig. 55. Data path of horizontal mode.....	54
Fig. 56. Coding order of residual blocks.....	55
Fig. 57. Transform matrix of 4x4 DCT transform .....	56
Fig. 58. Transform matrix of 4x4 IDCT transform.....	56
Fig. 59. Transform matrix of Hadamard transform.....	56
Fig. 60. Fast algorithms of 4x4 transform.....	56
Fig. 61. Hardware architecture of transform unit.....	57
Fig. 62. Hardware architecture of quantization unit.....	57
Fig. 63. Hardware architecture of mode decision unit .....	58
Fig. 64. CAVLC architecture .....	58
Fig. 65. Memory Organization.....	59
Fig. 66. Timing schedule of proposed intra coder.....	60
Fig. 67. Timing schedule of proposed architecture.....	60
Fig. 68 Chip specification .....	62



# Chapter 1 Introduction

The Advanced Video Coding (AVC) is the latest generation standard developed by a Joint Video Team (JVT) of ISO/IEC and ITU-T[1]. The new standard outperforms the earlier MPEG-4 and H.263 standards, providing better compression of video images. While the basic framework of H.264/AVC is similar to the motion compensated hybrid scheme of previous video coding standards, additional tools improve the compression efficiency at the expense of an increased implementation cost.

## 1.1. Motivation

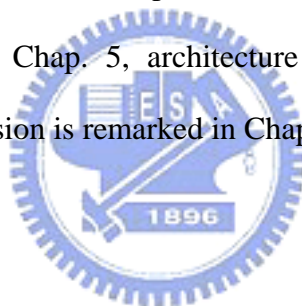
The high-efficient coding features of H.264/AVC are due to complex mode selection and high computational coding tools. For software implementation, H.264/AVC video coding demands fast algorithm to minimize the computation complexity for mode decision. To meet the need of consumer electronics market, VLSI implementation is necessary for real-time and low power applications. These motivate us to explore efficient solution for key modules in H.264/AVC.

Deblocking filter of H.264/AVC is both computational and memory intensive due to its highly adaptive mode decision and small 4x4 block sizes. The small 4x4 block size used in H.264/AVC requires almost every pixel in a frame loaded from and written to frame memory for deblocking operations. It is reported that even with highly optimized filtering algorithm, the deblocking operation still occupies one third of the computational complexity of a decoder. In order to solve the problem mentioned, two architectures are proposed to meet the high resolution real-time deblocking filter process.

Intra prediction is the dominate components besides the motion estimation in the encoding process. Exhaustedly search is adopted in the reference software to select the optimal intra prediction mode. Since each mode will be examined, the computation load is quite large and becomes the one of computational bottleneck. A fast intra prediction algorithm is needed to speed up the encoding process.

## **1.2. Thesis Organization**

This thesis contains six parts. Chap. 1 gives the motivation and design challenge of this work. In Chap. 2, a brief overview is given for H.264/AVC coding standard. Then, the proposed deblocking architectures and their cost-performance analysis are presented Chap. 3. In Chap. 4, a fast three step intra prediction algorithm is contributed. In Chap. 5, architecture design for intra coding is implemented. Finally, conclusion is remarked in Chap. 6.

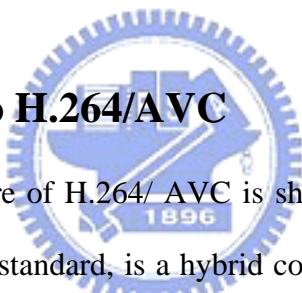




## Chapter 2 Overview of H.264/AVC Standard

In the recent years, multimedia application becomes more flexible and more powerful with the development of digital signal processing and communication technology. The Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG develop a new standard for the compression of natural video images. The new standard [1][2] is known as H.264 and also MPEG-4 Part 10 Advanced Video Coding, and regarded as the next generation video compression standard. The new standard is designed for technical solutions of wide application areas from videoconference, broadcasting, digital storage media, multimedia streaming service, etc.

### 2.1. Introduction to H.264/AVC



The overall architecture of H.264/AVC is shown in Fig. 1. , the same with the previous video coding standard, is a hybrid coder. Different from prior video coding standards, H.264/AVC has many features that enhance coding efficiency to predict the content of a picture.

- Variable block-size motion estimation/compensation

As shown in Fig. 2, H.264/AVC has more flexibility in selection of block sizes and shapes, such as 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4.

- Quarter-sample-accurate motion vector accuracy

Compare to advanced profile of the MPEG-4 Visual standard, 6-tap filter is adopted in H.264/AVC to reduce the complexity of interpolation.

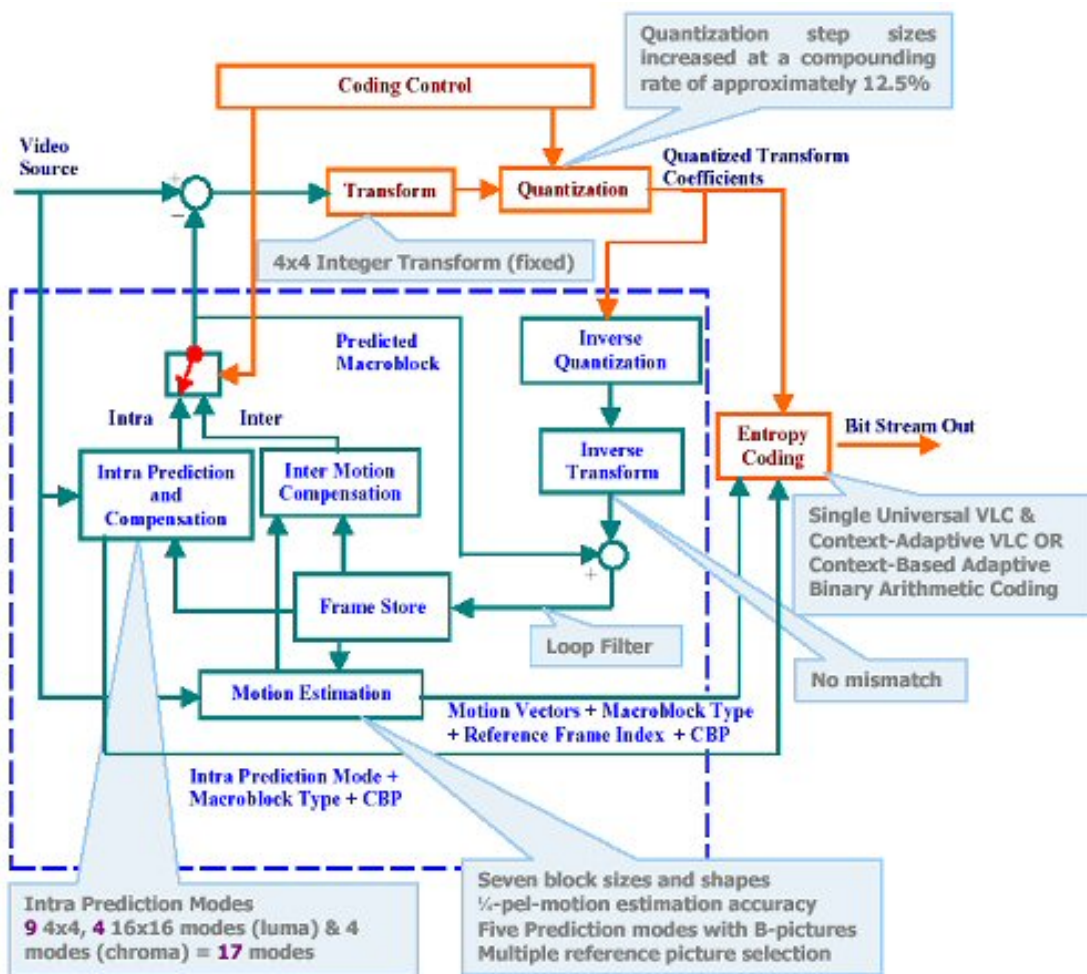
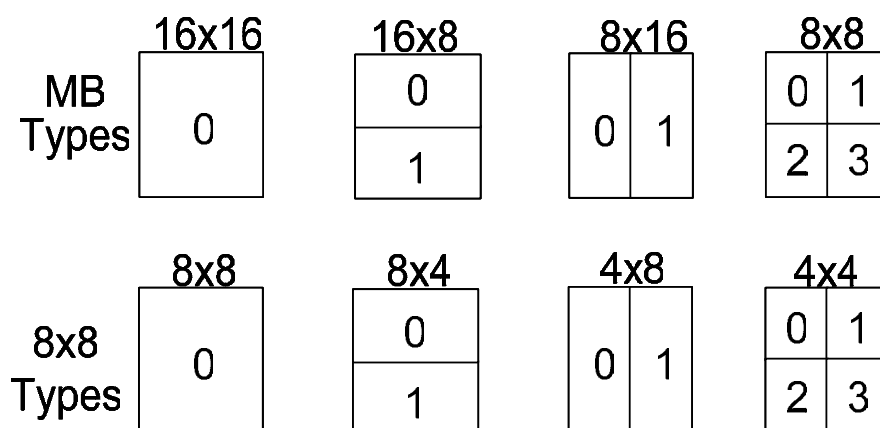


Fig. 1 Block diagram of H.264/AVC encoder



**Motion vector accuracy 1/4 (6-tap filter)**

Fig. 2 Block size of motion estimation/compensation

- Multiple reference picture motion estimation/compensation

H.264/AVC adopts the multiple reference picture selection technique to enable efficient coding by allowing an encoder to select the reference frame. There are at most five previous and five afterward reference pictures to be searched.

- Directional spatial prediction for intra coding

In H.264/AVC intra encoding, the edges of the previously decoded sample of current picture is applied to predict the samples of current block to be encoded. In summary nine kinds of 4x4 luma prediction modes, four kinds of 16x16 luma prediction modes, and four kinds of 8x8 chroma prediction modes are adopted.

- Small block size integer transform

Due to small block size motion estimation/compensation, H.264/AVC standard is based primarily on the 4x4 transform, including discrete cosine transform and discrete hadamard transform. It requires only 16 bits arithmetic processing.

- In-loop deblocking filter

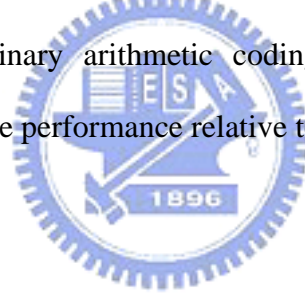
The block-based video coding produces blocking artifact due to its block structure. Blocking artifact becomes worse especially in the low bit rate or highly compressed video environment. To reduce the artifact, the in-loop deblocking filter is adopted by the H.264/AVC standard to improve the quality of decoded picture. Fig. 3 shows the subjective view comparison of picture with deblocking filter and without deblocking filter.



Fig. 3 Subjective view comparison of picture with deblocking filter (left) and without deblocking filter (right).

- Context-adaptive entropy coding

The two entropy coding methods applied in H.264/AVC, termed CAVLC (context-adaptive variable length coding) and CABAC (context-adaptive binary arithmetic coding), both use context-based adaptivity to improve performance relative to prior standards.



- CABAC

In main profile, an advanced entropy coding method known as arithmetic coding is included in H.264/AVC to increase the efficiency of entropy coding.

With all the mentioned powerful coding approaches and extensive rate distortion optimization (RDO) techniques, H.264/AVC can offer a significant improvement of bit-rate reduction compared with previous video standards under the same PSNR quality as shown in Fig. 4. It is reported that the new standard can achieve 39%, 49%, 64% of bit-rate reduction compared with MPEG-2[3], H.263[4], and MPEG-4[5] respectively[6]. However, the complexity and computation load of video coding in H.264 increase drastically.

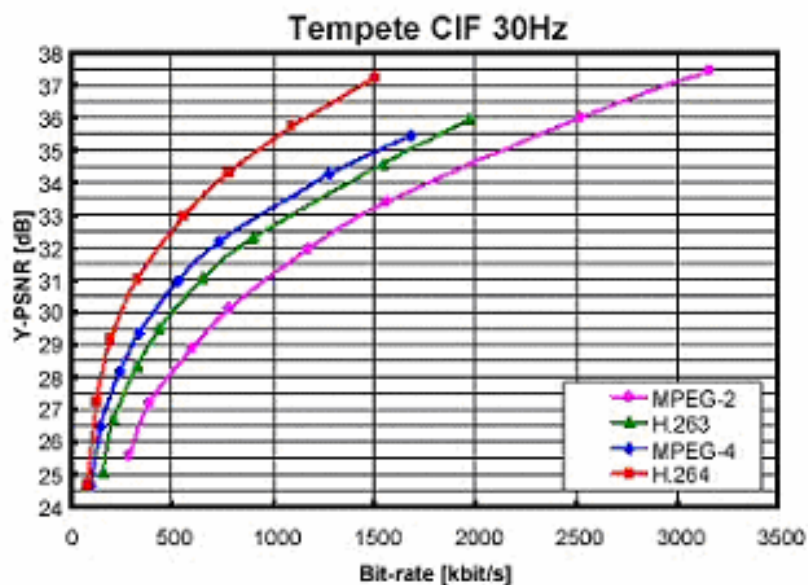


Fig. 4 R-D curve comparison of H.264/AVC with MPEG-4, H.263, and MPEG-2

## 2.2. Profile and Level

In H.264/AVC, three profiles are defined, which are the Baseline, Main, and Extended Profile as shown in Fig. 5. The Baseline profile supports all features in H.264/AVC except the following two feature sets:

- Set 1: B slices, weighted prediction, CABAC, field coding, and picture or macroblock adaptive switching between frame and field coding.
- Set 2: SP/SI slices, and slice data partitioning.

The first set of additional features is supported by the Main profile. However, the Main profile does not support the FMO, ASO, and redundant pictures features which are supported by the Baseline profile. Thus, only a subset of the coded video sequences that are decodable by a Baseline profile decoder can be decoded by a Main profile decoder.

The Extended Profile supports all features of the Baseline profile, and both sets of features on top of Baseline profile, except for CABAC.

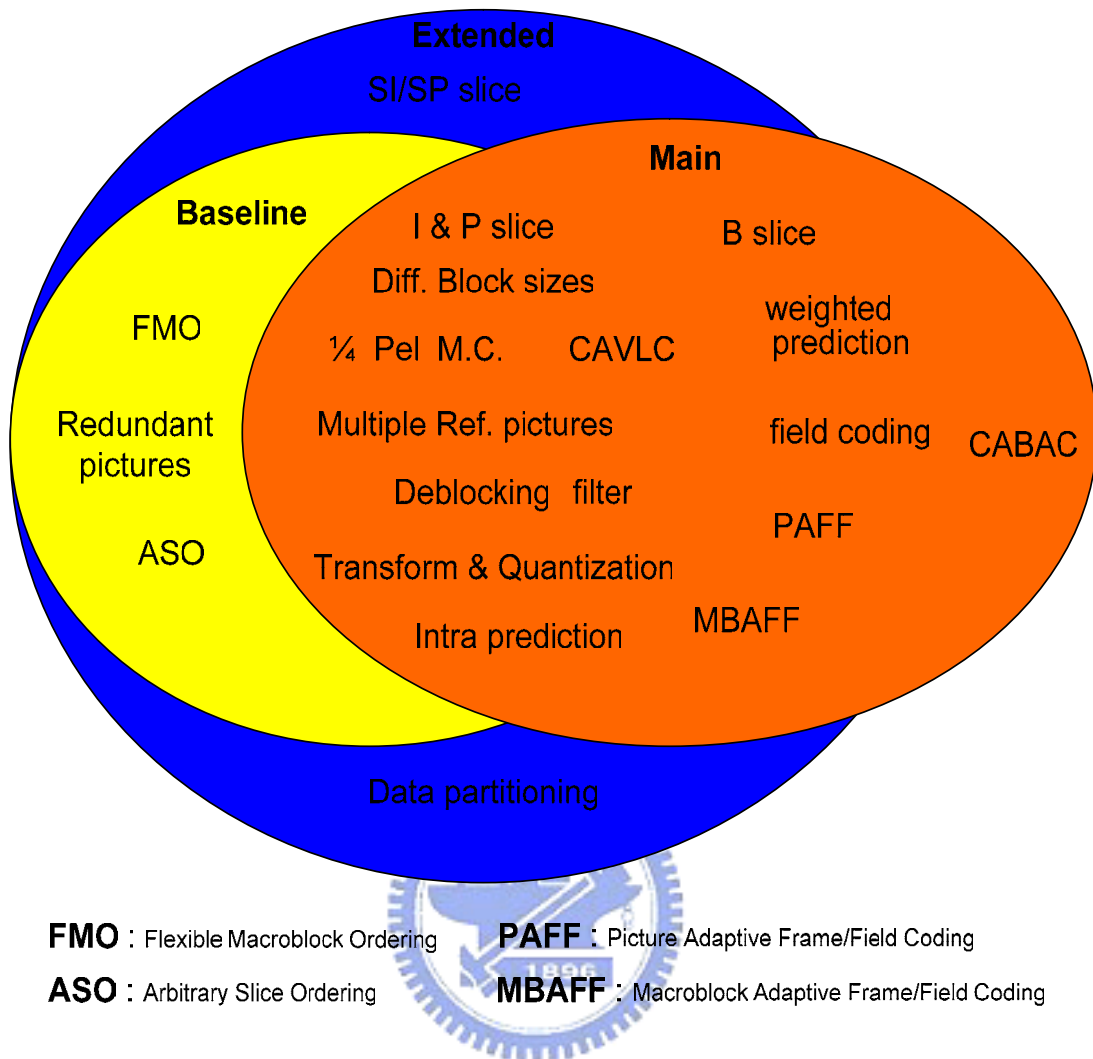


Fig. 5 Profile of H.264/AVC

In H.264/AVC, the same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile. There are 15 levels defined, specifying upper limits for the picture size (in macroblocks) ranging from QCIF to all the way to above 4k 2k, decoder-processing rate (in macroblocks per second) ranging from 250k pixels/s to 250M pixels/s, size of the multipicture buffers, video bit rate ranging from 64 kbps to 240 Mbps, and video buffer size.

## Chapter 3 Architecture Design of Deblocking

### Filter in H.264

In the H.264/AVC standard, the adaptive deblocking filter is applied on edges of each 4x4 blocks in a macroblock (MB) to reduce the blocking artifact. However, the deblocking filter is both computational and memory intensive due to its highly adaptive mode decision and small 4x4 block size [9]. The adaptive mode decision is required for each edge to distinguish real edges from block artifacts. The small 4x4 block size used in H.264/AVC requires almost every pixel in a frame loaded from and written to frame memory for deblocking operations. It is reported that even with highly optimized filtering algorithm, the deblocking operation still occupies one third of the computational complexity of a decoder [9]. Thus, VLSI implementation is necessary for real-time and low power applications.

In this chapter, two deblocking filter architectures are proposed. For the first version, the data flow is reordered for easy and regular hardware implementation while maintains the standard compatibility. For the second version, an in-place computing design for the deblocking filter is presented. The proposed in-placed computing flow reuses intermediate data to filter horizontal edges and vertical edges seamlessly as soon as data is available. Thus, the intermediate data storage is greatly reduced to only the four 4x4 blocks instead of whole 16x16 macroblock. In the first version, gate count is greatly reduced by simple control unit, and internal memory is also reduced to 50% of that in the previous design. In the second version, the proposed architecture can reduce 90% of internal memory and achieve higher throughput than others.

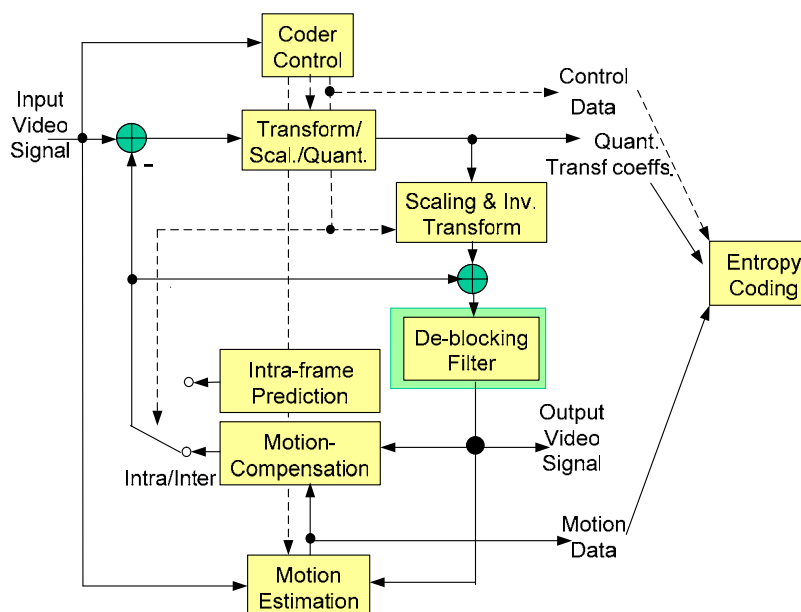


Fig. 6 Encoding loop of H.264

Both of them are implemented by UMC 0.18 $\mu$ m CMOS technology. The resulting hardware of Version 1 can achieve real-time 2Kx1K (2048x1024) 30Hz video at 82.58 MHz. The gate count is only 9.16K when synthesized at 100MHz, excluding the memory cost. For version 2, the resulting hardware can achieve real-time 2Kx1K (2048x1024) 30Hz video at 73.73 MHz. When synthesized at 100MHz the gate count is only 13.41K, excluding the memory cost.

### 3.1. Fundamental of H.264/AVC Deblocking Filter

The block-based video coding, due to its simple and regular block structure, has been widely used in various video standards, such as MPEG-1, MPEG-2, MPEG-4 and H.26x. However, block-based computation like discrete cosine transforms (DCT) and motion compensation (MC) also produce blocking artifact [7][1][8][9], which becomes worse especially in the low bit rate or highly compressed video environment. To reduce blocking artifact, the deblocking filter is a well-known tool to improve both objective and subjective video quality, either inside or outside the coding loop. In-loop approach is adopted by the



H.264/AVC standard as shown in Fig. 6. The in-loop deblocking filter improves the quality of reference frame, thus improves overall resulting view effect. However, This forces all standard conformant decoders to perform identical filtering in order to stay in synchronization with the encoder.

H.264/AVC deblocking filter is adaptive on several levels. On the slice level, the global filtering strength can be adjusted to the individual characteristics of the video sequence. On the block-edge level, filtering strength is made dependent on the inter/intra prediction decision, motion differences, and the presence of coded residuals in the two participating blocks. Special strong filtering is applied for macroblocks with very flat characteristics to remove “tiling artifacts”. On the sample level, sample values and quantizer-dependent thresholds can turn off filtering for each individual sample.

Deblocking process is done in MB by MB in raster scan order. In each MB, the processing order in the H.264/AVC reference software first processes on the four vertical edges from left to right, transposes the intermediate data, and then processes on the four horizontal edges from up to bottom, as shown in

Fig. 7.

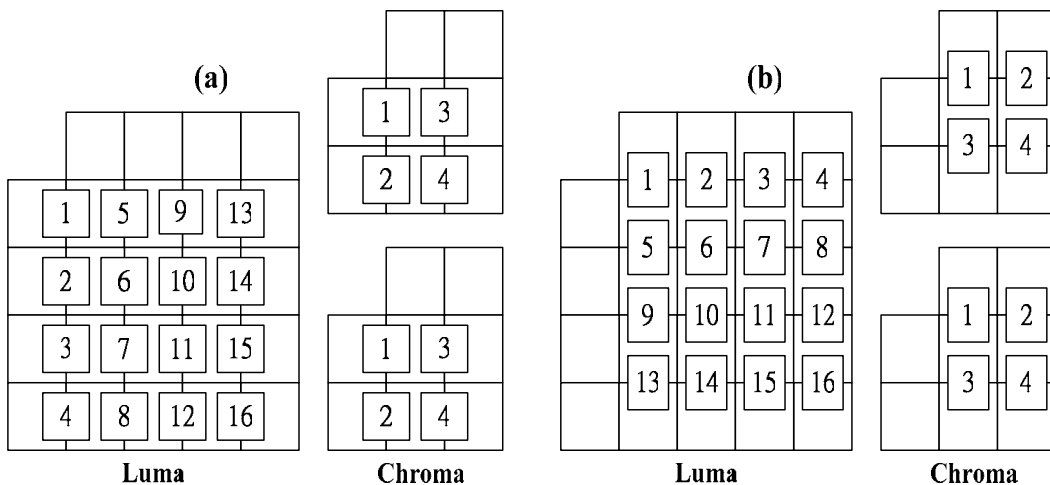


Fig. 7. Original processing flow for (a)horizontal filtering, and (b)vertical filtering

Table 1 Parameters for determining boundary strength

Block modes and conditions	Bs
One of the blocks is Intra and the edge is a macroblock edge	4
One of the blocks is Intra	3
One of the blocks has coded residuals	2
Difference of block motion $\geq 1$ luma sample distance	1
Motion compensation form different reference frames	1
Else	0

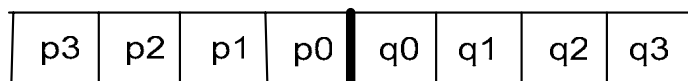


Fig. 8 Convention for describing samples across two 4x4 block boundary.

The Boundary-Strength (Bs) parameter, a number ranging from 0 to 4, is assigned to every boundary between two neighboring 4x4 luma sample blocks to determine whether it is true blocking artifact or not. The chroma boundary strengths are the same as that in the corresponding luma boundary location.

Table 1 shows that the value of Bs depends on the modes and the coding conditions of the two adjacent blocks. In this table, conditions are evaluated from top to bottom until one of the conditions holds true, and the corresponding value is assigned to Bs. Bs decides the filter strength performed on the edge. Two primary filtering modes are selected. A value of 4 means the strongest filtering mode, and the Bs from 1 to 3 is the standard mode, whereas a value of 0 means no filtering is applied on the edges.

For nonzero Bs values, a pair of quantization dependent parameters, referred to

as  $\alpha$  and  $\beta$ , are used to determine which set of samples to be filtered. In the following description, the convention for describing 8 pixels across two 4x4 block boundary is shown in Fig. 8. Filtering on a line of 8 samples takes place if the three conditions

$$|p_0 - q_0| < \alpha, |p_1 - p_0| < \beta, |q_1 - q_0| < \beta, \text{ when } B_s \neq 0$$

hold true. For edges with  $B_s$  from 1 to 3, the filter operation is divided into basic filter operation and clipping. In strongest filtering mode ( $B_s=4$ ), the deblocking operation uses a very strong 4- and 5-tap filter that modifies the edge sample and two interior samples on each side, or uses a weaker 3-tap filter to modify the edge samples only. The stronger filter is only applied when the following constraint

$$|p_0 - q_0| < (\alpha \gg 2) + 2$$

holds true. Interested readers can refer to [1] and [10] for more details.

## 3.2. Architecture Design of H.264/AVC Deblocking Filter

### 3.2.1. Version 1

#### 3.2.1.1. Simple Data Flow

The major drawback of this direct approach is that intermediate data between different edges has to be stored and loaded again. Thus results in an inefficiency data flow and complex controller. For an efficient VLSI design of the deblocking filter, regular data flow is the major concern for easy hardware implementation. Different from the original processing order that processes the column major order first, the proposed computing flow processes the horizontal filtering along the row major order first and then vertical filtering along the column major order as shown in Fig. 9.

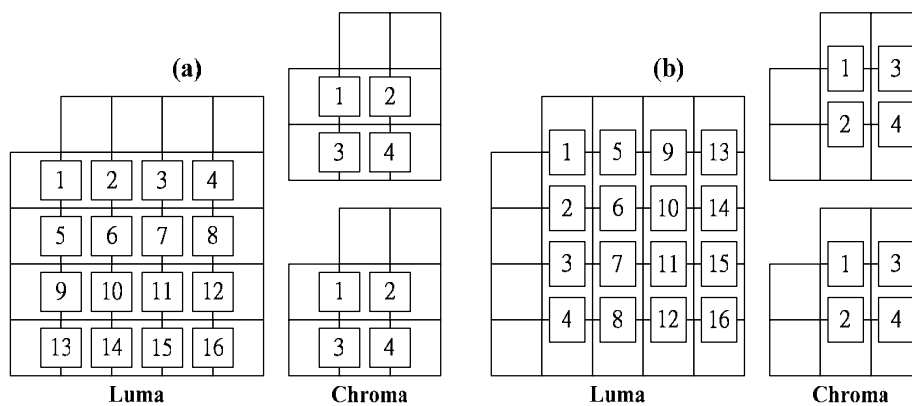


Fig. 9. Modified processing flow for (a) horizontal filtering, and (b) vertical filtering

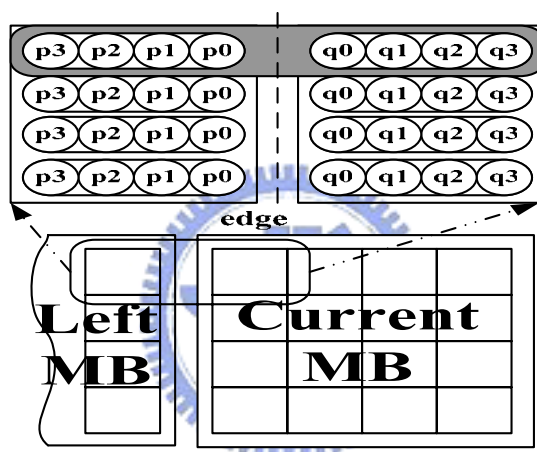


Fig. 10. Data structure of deblocking filter

With the modified approach, the intermediate 4x4 block data between neighboring vertical or horizontal edge will be reused immediately. Thus we can save the internal memory access and speed up the deblocking computation. This modification does not only fit the memory access order but also has higher data reuse capability and still has the same results as the standard specified.

### 3.2.1.2. Hardware Architecture

The data structure of proposed deblocking filter is a line of 8 pixels as shown in

Fig. 10. The proposed architecture is shown in

Fig. 11. A 1-D 8 pixels parallel-in parallel-out deblocking filter can be reconfigurable to support different filtering strength. A 4x4-pixel shift register is to reuse the intermediate 4x4 block data after processing previous neighboring edge. A 4x4-pixel transpose register is to transpose data from row major order to column major order, or transpose from column major order back to row major order after vertical filtering. An 80x32bits SRAM is required to buffer the intermediate 20 4x4-block pixels to be filtered vertically.

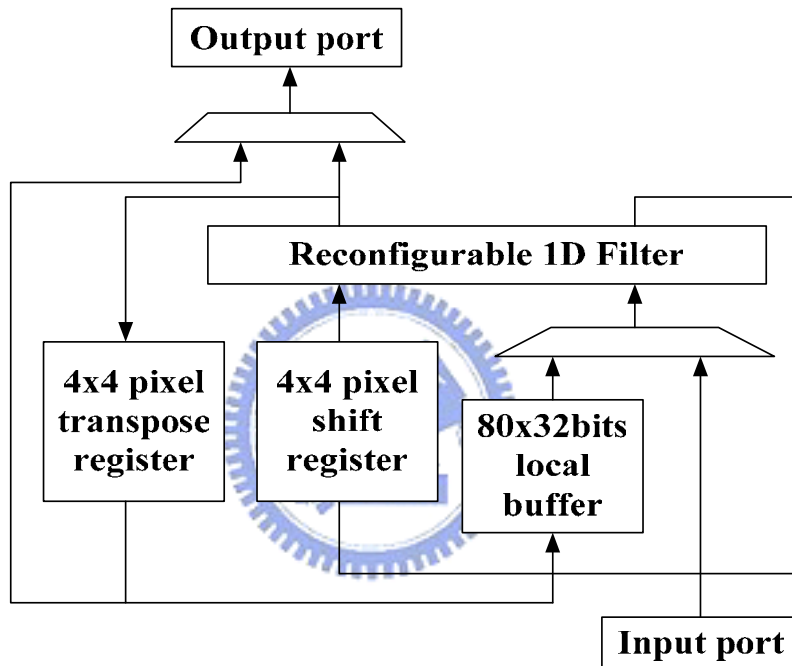


Fig. 11. The proposed deblocking filter architecture.

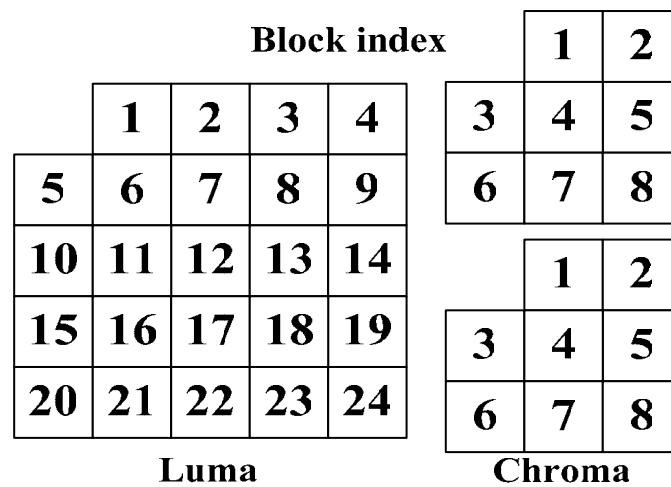


Fig. 12. 4x4 Block index for one macroblock.

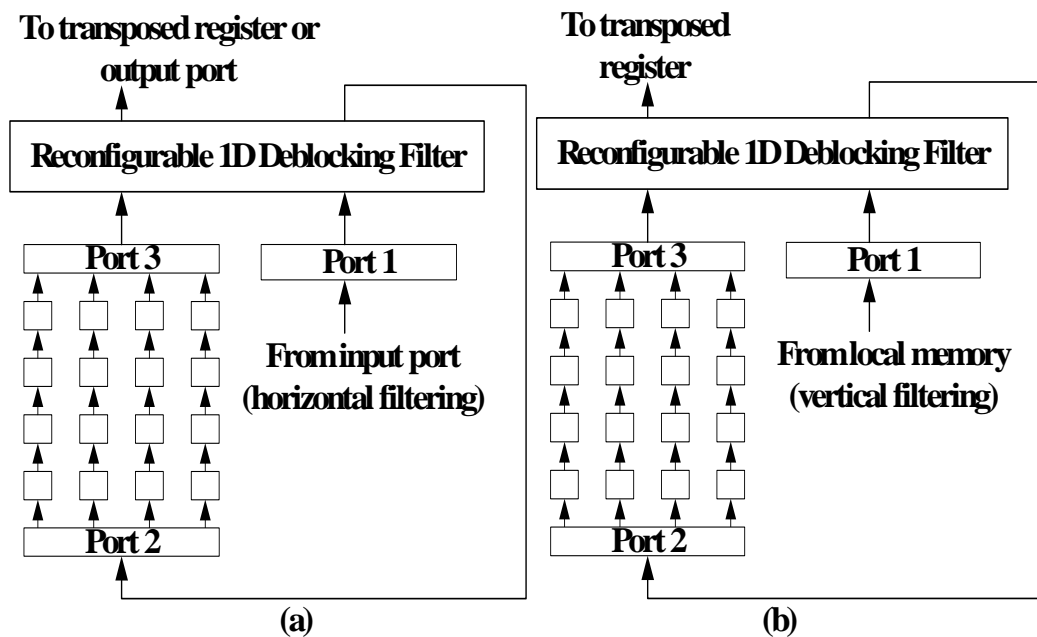


Fig. 13 Data path for (a) horizontal filtering and (b) vertical filtering

The proposed architecture operates as below. For simplicity of explanation, we use the block index as shown in Fig. 12 in the following. First, we assume that all data I/O is row major order with 32bits width (4 pixels in parallel). The data path is shown in Fig. 13.

#### Step 1: data preparation

In the first 16 cycles, we first retrieve the data of block index 1 to 4, transpose them from row major order to column major order with the transpose buffer, and store them in the local buffer for vertical filtering.

#### Step 2: horizontal filtering over vertical edges

Then we start horizontal filtering as the order shown in Fig. 9(a) with data of block 5 to 24. In this phase, filter inputs are from external memory via input port, and from the 4x4 shift register via port3. The data after first time filtering is sent to 4x4 shift register via port 2 to be reused to filter next neighboring edge.

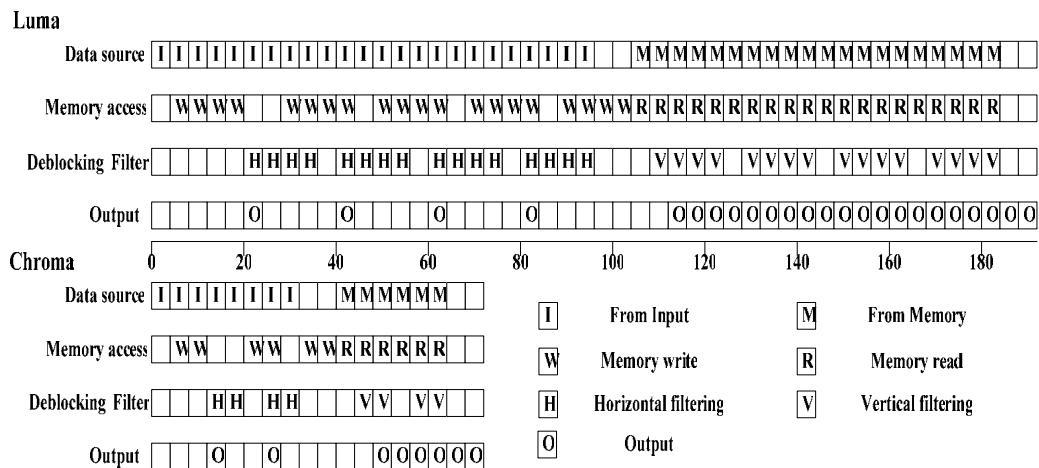


Fig. 14 timing diagram of simple data flow deblocking filter

Step 3: vertical filtering over horizontal edges

After the processing of horizontal filtering, the macroblock data is transposed to column major order by transpose buffer and stored in the local buffer for vertical filtering. The vertical filtering uses the same data flow as the horizontal filtering. The only difference is that filter input is from the local buffer instead of external memory. The filtered results are transposed again and stored back to the external memory in the row major order. Thus, all data uses row major order input and row major order output to the external memory. This can ease the other relating processing to work together.

Based on the above flow, it needs only  $192(Y)+72(Cb)+72(Cr)=336$  cycles to complete the deblocking process for one YCbCr macro block. The processing timing diagram of proposed deblocking is shown in Fig. 14 that illustrates the detailed timing of above steps.

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>
<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>
<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>

Fig. 15. Block index

## 3.2.2. Version 2

### 3.2.2.1. Full Data Reuse Flow

In version 1, the deblocking filter always reuses the data of neighbor 4x4 blocks of current filtering edge. After horizontal filtering, there are two edges can be filtered in the next step. Those are up edge and right edge. In version 2, the data flow is further improved to explore more data reusability

Fig. 15 shows the Block index for explanation. Block 1-4 are the intermediate data from above macroblock after vertical edge processing, and block 5, 10, 15, and 20 are the intermediate data from left macro block after horizontal edge processing. For simplicity, we will denote the block number as  $blk_i$ , where  $i$  is from 1-24. These block data will be processed with current macro block data to complete the deblocking operations.



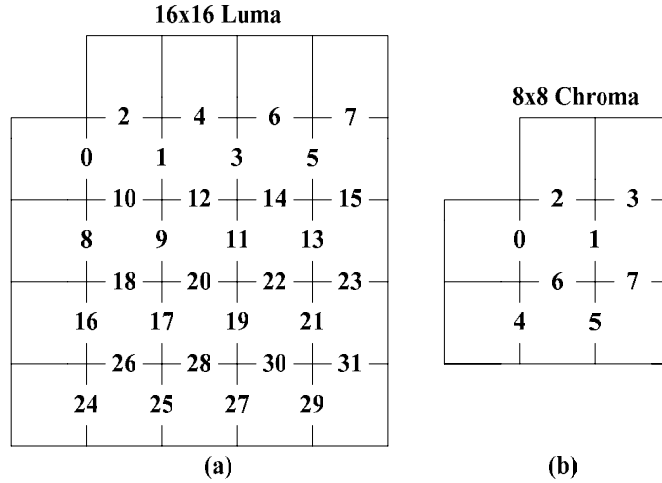


Fig. 16. Edge processing order for (a) luma edge, and (b) chroma edge

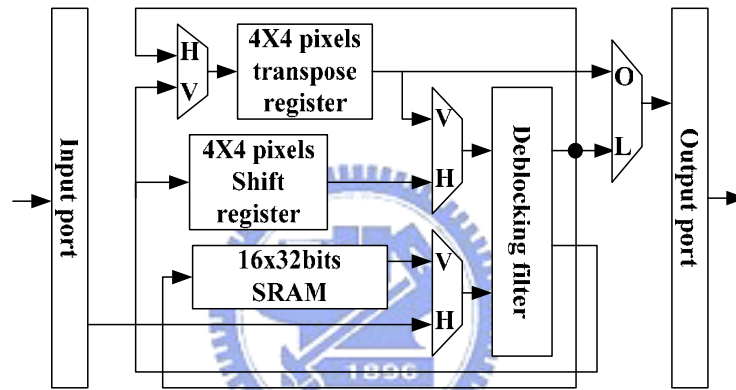


Fig. 17. Overall architecture

Fig. 16 shows the proposed full data reuse flow that maintain the same result as specified by the H.264/AVC standard. As shown in Fig. 16 (a), we process edges of each 4x4 block from the left-top most block (blk6) to the right-bottom most block (blk24). Starting from the left-top most 4x4 block (blk6), we first do the horizontal filtering over its two vertical edges (edge 0 and edge 1). Then, since all data is available for horizontal edge 2 (intermediate data from blk1 and blk6), we can do the vertical filtering over the top horizontal edge (edge 2). This horizontal-vertical interleaved approach is repeated for each 4x4 block in a raster scan order, as the edge number shown in Fig. 16 (a) and (b).

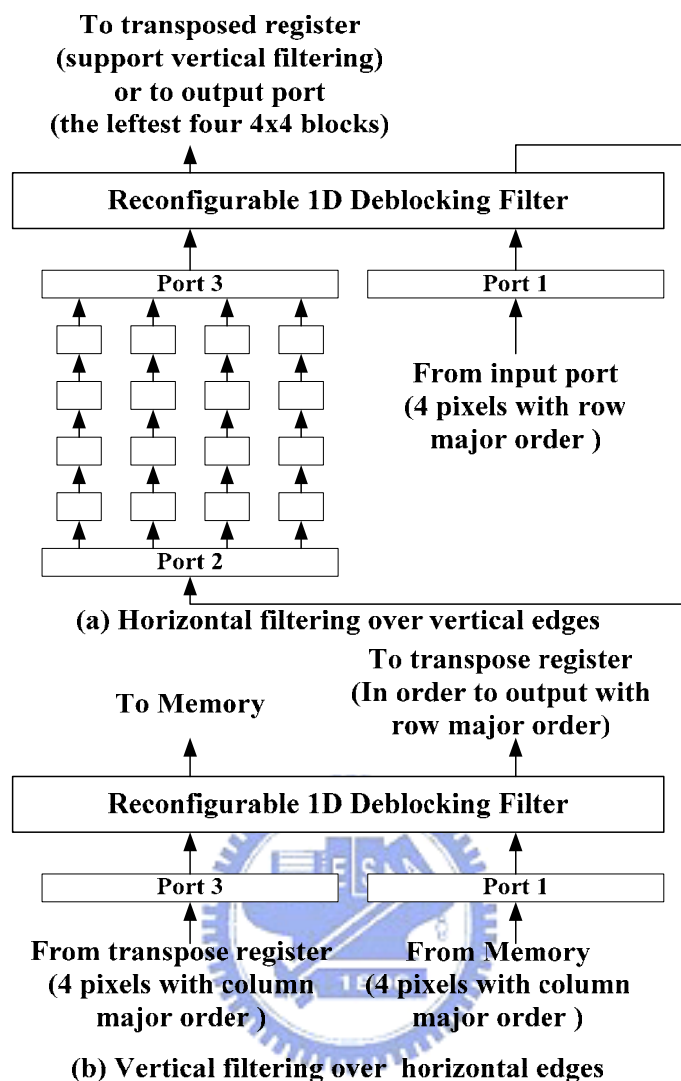


Fig. 18. Data path (a) horizontal filtering over vertical edges, and (b) vertical filtering over horizontal edges

With the interleaved approach, the intermediate data will be reused immediately. Thus we can save the memory access and buffer required to process the left, top, and right edge in a 4x4 block. The only buffer and memory access remained are the intermediate data for the bottom edge in a 4x4 block. Therefore, we need only four 4x4 blocks (4x16x8bits) above the current filtering block row (e.g. store blk6-blk9 when process blk11-blk14), rather than a whole macroblock (24x16x8bits) as in the conventional data flow. Because of such high data reuse, internal memory access number and size are both greatly reduced.

### 3.2.2.2. In-place Deblocking Filter Architecture

Fig. 17 shows the proposed architecture, where the solid arrows denote 32-bits dataflow. First, we assume that all data I/O is row major order with 32bits width (4 pixels). For simplicity, we will only explain the operation of luma macroblock. The chroma block is processed using the same method.

In the first 16 cycles, the data of block index 1 to 4 are transposed from row major order to column major order with the transpose buffer, and store them in the local SRAM buffer to wait for vertical filtering. Then we start horizontal filtering over the vertical edge 0 as shown in Fig. 16 (a) with data of block 5 and 6. The data of block 5 is from external memory via input port and shifted into to 4x4 shift register after four cycles. After that, the data of block 6 from input port, and data of block 5 from 4x4 shift register is loaded to perform the horizontal filtering as shown in Fig. 20 (a). The filtered data of block 5 is sent to output port, and the data of block 6 after first time filtering is sent to 4x4 shift register. So the data in 4x4 shift register can be used to perform the next horizontal filtering again.

Next we start horizontal filtering over vertical edge 1 with data of block 6 and 7. In this phase, filter input is from 4x4 shift register (block 6), and from external memory (block 7) via input port as shown in Fig. 20 (b). The filtered data of block 6 is then transposed to column major order by transpose buffer after the data is being filtered two times, and the data of block 7 after first time filtering is sent to 4x4 shift register.

The data of block 1 and block 6 are both ready to perform vertical filtering over horizontal edge 2 in column major order now. In this phase, filter input is from local SRAM buffer (block 1), and from 4x4 transpose register (block 6) as shown in Fig. 20 (c). The data of block 6 after filtering is sent to local SRAM

buffer to wait for filtering next vertical edge, and the data of block 1 is transposed again and output in row major order in the next four cycles. The remaining edges is processed using the same data flow mentioned above.

Same as version 1, this design use parallel-in parallel-out style (32-bits, processing 4-pixels concurrently). The deblocking filter part implements the required function as specified by the H.264/AVC standard. The 16x32bits SRAM buffers four 4x4 block pixels to be processed, as described in the previous Section. The register array is for transposing operation during the 2-D deblocking filtering.

### 3.2.2.3. Memory Organization

In the proposed data flow, the deblocking operation uses horizontal-vertical interleaved scheduling. However, to support such the interleaved operation, the corresponding architecture shall transpose the 4x4 block immediately when changing the filtering edges. The corresponding data path consists of a 4x4 shift buffer and a 4x4 transpose buffer. We assume that the SRAM module is an ordinary one which has one 32bits read port and one 32bits write port. With this, we can transpose the data of 4x4 blocks to support both horizontal filtering and vertical filtering on a parallel-in parallel-out deblocking filter seamlessly.

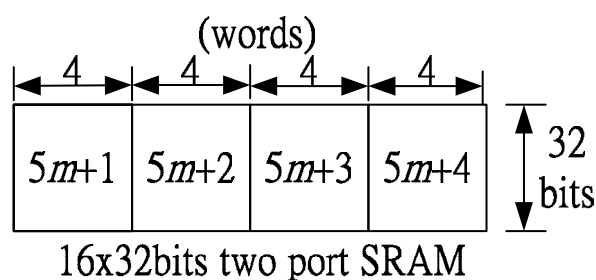


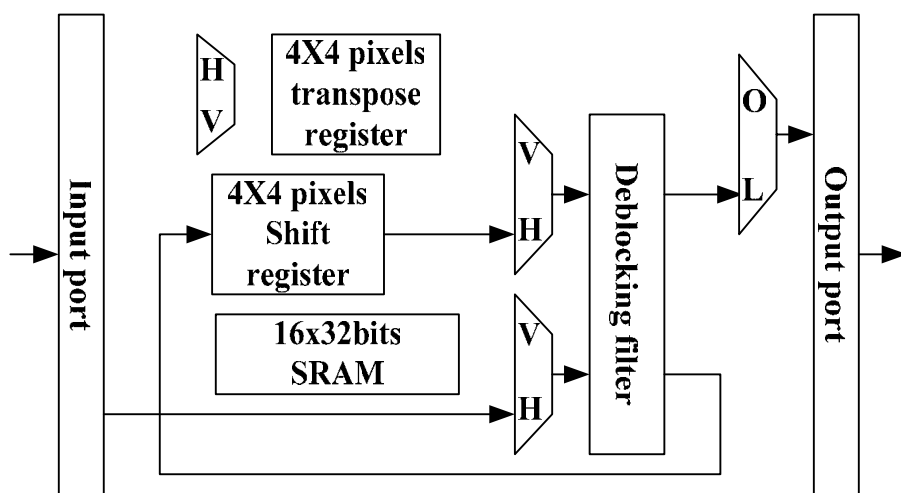
Fig. 19 Organization of on-chip 1R/1W port SRAM

The on-chip buffer first stores blk1 to blk4. Fig. 7 shows its data organization, where  $m$  is an integer number from 0, the number in the block denotes the block index and each block stores one  $4 \times 4$  block data. Then the same address location will be overwritten by new data from blk6 to blk9, respectively. This will be repeated for each row of  $4 \times 4$  blocks. Since data is in-place overwritten after reading out, no read-write conflict will occur.

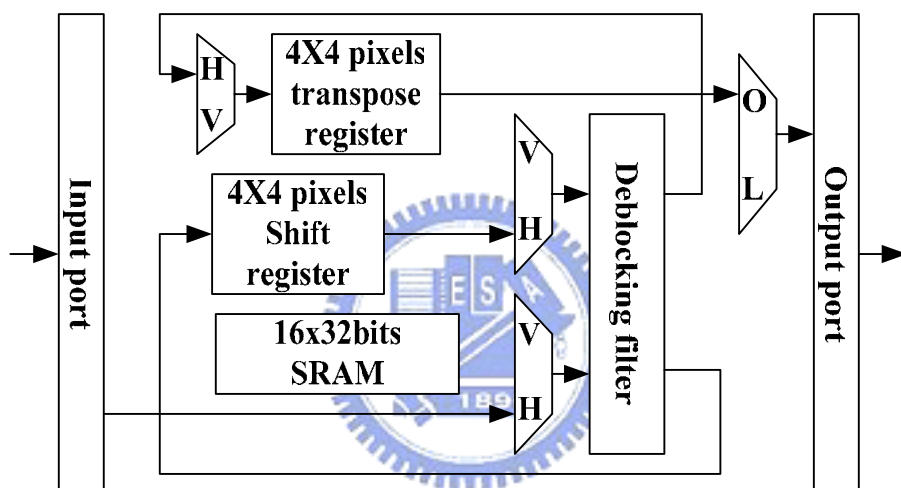
### 3.2.2.4. Processing Schedule

Assume input data are four pixels (32bits) per clock cycle, it requires  $16 + (36+4) \times 4 + 16 = 192$  cycles to process a luma macroblock without overlapping the data flow. Among them, 16 cycles to input data of block 1 to 4 from external memory to on-chip memory before starting processing the data, 36 cycles to process edge  $8m$  to  $8m+7$ , 4 cycles to shift block  $8m+7$  from  $4 \times 4$  shift buffer to  $4 \times 4$  transpose buffer, and 16 cycles to output data of block 21 to 24 from on-chip memory to external memory, where  $m$  is an integer number from 0. For two chroma macroblocks,  $(8 + (20+4) \times 2 + 8) \times 2 = 128$  cycles is required. Thus the processing capability is 320 cycles per macroblock.

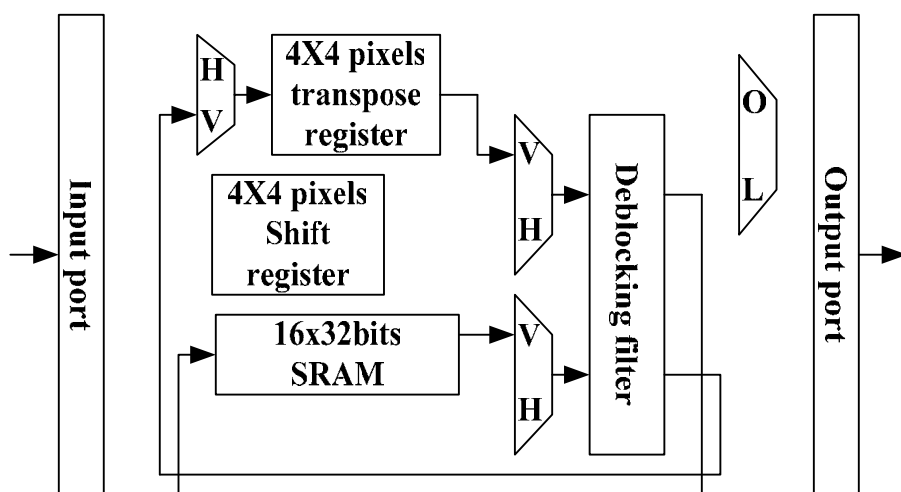
For a more efficient processing schedule, we can save four cycles by overlapping the data loading cycles of block  $5(m+1)$  from the external memory and data shifting cycles of block  $5m+4$  from  $4 \times 4$  shift buffer to  $4 \times 4$  transpose buffer before processing horizontal edge  $8m+7$ .



(a) Horizontal filtering over the left vertical edge 0, 8, 16, and 24



(b) Horizontal filtering over vertical edges



(c) Vertical filtering over horizontal edges

Fig. 20. Data flow of deblocking filter (a) processing the left vertical edges, (b) processing vertical edges, and (c) processing horizontal edges.

For example, it takes four cycles to shift the data of block 9 to 4x4 transpose register. At the same time, we can move the data of block10 from external memory to 4x4 shift register. With this strategy, the total cycle count is reduced to 300 cycles for one macroblock

### 3.3. Implementation and Comparison

To evaluate the accuracy and the efficiency of the proposed architecture, the proposed architectures are designed by Verilog and implemented by TSMC 0.18 $\mu$ m CMOS technology. The resulting hardware of Version 1 can achieve real-time 2Kx1K (2048x1024) 30Hz video at 82.58 MHz. The gate count is only 9.16K when synthesized at 100MHz, excluding the memory cost. The resulting hardware of version 2 can achieve real-time 2Kx1K (2048x1024) 30Hz video at 73.73 MHz. The gate count is only 13.41K when synthesized at 100MHz, excluding the memory cost.

Table 2 lists the area cost comparisons with other approaches. Table 3 shows the processing capability comparison. Table 4 shows the comparison of memory size. From the comparison results, the proposed architecture has the advantages of both smaller area cost and cycle count because of high data reusability.

Table 2. Comparison of cost synthesized at 100MHz. (excluding memory cost).

Design	Gate count
Version 1 (with single port SRAM)	9.16K
Version 2 (with 1R/1W port SRAM)	13.41K
[4] Basic type (with single port SRAM)	18.91K
[4] Advanced type (with dual port SRAM)	18.91K
[4] Basic type (with two port SRAM)	18.91K
[4] Dual arrays type (with two port SRAM)	20.66K

Table 3. Comparison of memory size

Design	Version 1	Version 2	[4]
Memory size	80x32bits	16x32bits	160x32bits

Table 4. Comparison of processing capability

Design	Cycle /MB	CIF (352X288)	2Kx1K (2048X1024)
Version 1 (single port SRAM)	336	3.99MHz	82.58MHz
Version 2 (1R/1W port SRAM)	300	3.56MHz	73.73MHz
[11] Basic type (single port SRAM)	878	10.43MHz	215.78MHz
[11] Advanced type (dual port SRAM)	814	9.67 MHz	200.05MHz
[11] Basic type (two port SRAM)	782	9.29 MHz	192.18MHz
[11] Dual arrays type (two port SRAM)	614	7.29 MHz	150.90MHz

### 3.4. Summary of Proposed Architectures

In this chapter, we contribute two high data reuse deblocking processing flow and its corresponding VLSI architecture for deblocking filter in H.264/AVC. By rearranging the data flow we can achieve high data reusability. Version 1 has very simple data flow, and a simple controller. In version 2 the major idea is to filter a vertical edge immediately followed by the filtering of a horizontal edge for a 4x4 block instead of whole macroblock. With a 4x4 transpose buffer, the aforementioned interleaved vertical and horizontal deblocking filtering can be easily realized. Thus, the processing capability of the proposed architecture can operate at high utilization and small memory size.



## Chapter 4 Fast 4x4 Intra Prediction Algorithm for H.264/AVC

Different from AC/DC prediction in MPEG-4, H.264/AVC adopts a new tool called intra prediction for intra frame coding. Intra prediction uses the directional spatial information to predict the sample to be encoded. However, intra prediction is also computational intensive besides the motion estimation in the coding loop. Direct approach for intra prediction use the full search that exhaustedly searches all possible modes and is adopted in the reference software. Although full search can achieve optimal prediction mode selection, it is computationally expensive. Besides, intra prediction is computed for intra-frame as well as inter-frame to determine the block type. It is thus highly desirable to develop fast intra-prediction mode selection.

In this chapter a fast algorithm for H.264 4x4 intra prediction is proposed. To determine the prediction mode, only six modes is required instead of nine modes in the full search method. The fast intra prediction algorithm can save 33% computational complexity with only about 1% bit-rate loss. Besides, the decision method is very simple.

### 4.1. Fundamental of H.264/AVC 4x4 Intra Prediction

There are 9 kinds of intra prediction modes for 4x4 intra blocks as shown in Fig. 21. A prediction mode is a way to generate 16 predictive pixel values (named a to p) using some or all of the neighboring pixels A to M as shown in Fig. 22. The pixels A to M are from the neighboring reconstructed blocks.

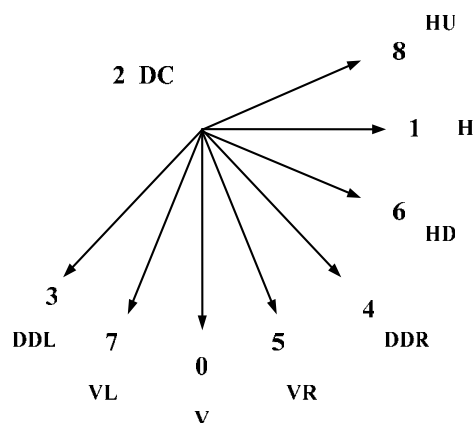


Fig. 21. Direction of 9 4x4 intra prediction modes in H.264

<b>M</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>
<b>I</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>				
<b>J</b>	<b>e</b>	<b>f</b>	<b>g</b>	<b>h</b>				
<b>K</b>	<b>i</b>	<b>j</b>	<b>k</b>	<b>l</b>				
<b>L</b>	<b>m</b>	<b>n</b>	<b>o</b>	<b>p</b>				

Fig. 22. A 4x4 block and its neighboring pixels

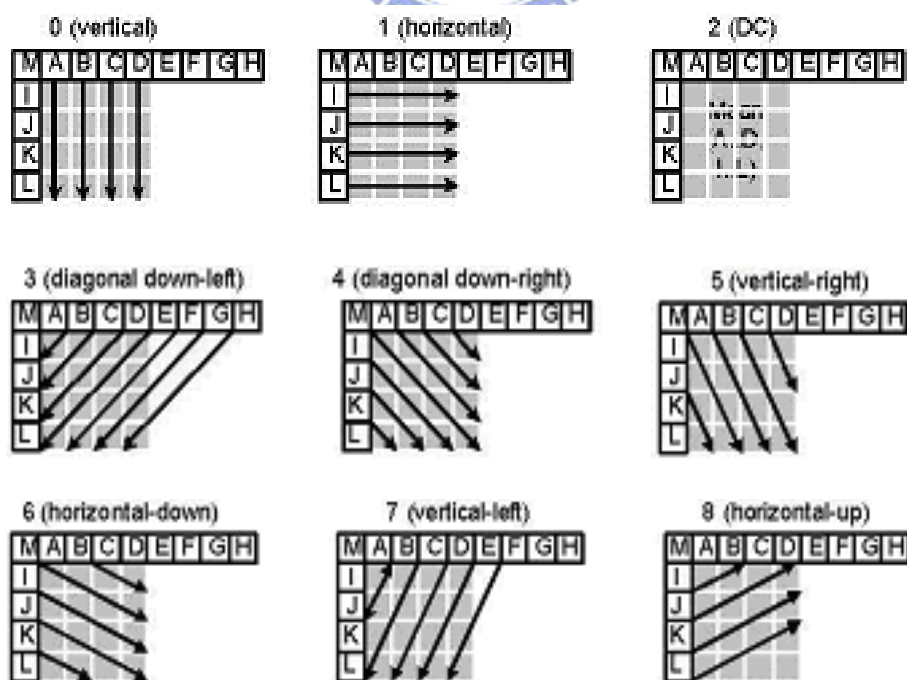


Fig. 23. 9 mode of 4x4 intra prediction

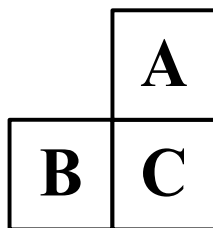


Fig. 24. Adjacent block of current 4x4 block

Fig. 23 shows the nine prediction modes designed in a directional manner. Mode 2 is called DC prediction in which all pixels (a to p) are predicted by  $(A+B+C+D+I+J+K+L)/8$ . Mode 0 is the vertical prediction mode in which pixels a, e, i, and m are predicted by A. Mode 1 is the horizontal prediction mode in which pixels a, b, c, and d are predicted by I. The other modes are similar except that the directions are different.

In addition to the 9 modes of 4x4 intra prediction, the correlation between spatially adjacent blocks is also exploited to encode the prediction mode efficiently. In Fig. 24, A and B are the up and left encoded 4x4 blocks near current block C. The probability of the 9 modes being the optimal intra prediction mode for C is different depending on the prediction modes of the top block A and left block B. A probability list is generated by Joint Video Team for each combination of the modes of A and B. Rather than sending the selected mode number, the position of the selected mode in the probability list is sent. Thus, the coding efficiency of intra prediction can be further improved.

In the reference software [13], a full search (FS) approach is used to examine all the 9 modes exhaustively to find the one with the smallest cost.

The main steps are:

1. Generate a 4x4 predicted block according to a mode

2. Calculate sum of absolute difference (SAD when using Hadamard is off) or sum of absolute transformed difference (SATD when using Hadamard is on) between the original 4x4 block and the predicted block

3. Compute Cost of the mode

$$\text{Cost} = \text{Cost\_of\_Mode} + \text{SAD (or SATD when using Hadamard is on)}$$

4. Repeat 1 to 3 for all the 9 modes, and choose the one that has the minimum cost.

Since each 4x4 block will go through these steps, the computation load is quite large and becomes one of computational bottleneck.

## 4.2. Review of Previous Approaches

Some approaches have been proposed on fast intra prediction algorithm [14][15][16]. In [3][4], they proposed a threshold to early terminate the computation of the most probable mode. If the cost of most probable mode is larger than the threshold, the quartet cost of remaining 8 modes is computed. The mode with minimum quartet cost is chosen among 8. Thus, the required computation cost varies with the contents of video. Besides, their algorithm still needs to determine a threshold, which could affect its efficiency severely.

In [16], it assumes high correlation between the edge direction and intra mode, and thus adopts the edge direction to predict the possible mode. However, the assumption of edge direction is not always true. Due to above drawbacks, these previous approaches increase the bit-rate significantly with PSNR loss. For example, approach in [16] increases bit- rate by 4.0% with 0.27dB PSNR loss in all I-frame condition.

### 4.3. Fast Three Step Intra Prediction Algorithm

In this chapter, a three step algorithm of intra prediction for intra 4x4 blocks is presented. The modes needed to be examined are constant 6 modes rather than variable number of modes as in the previous approaches.

From our observation, we find the SAD of the modes at the neighborhood of the optimal mode is also small. It means that we can skip some mode after initial search to save computation power. After initial search of some modes, we can examine the two modes neighboring to the selected mode in the first step. In the last step, extra one mode is refined to improve the prediction precision.

Fig. 25 shows a flow chart of the algorithm. We first start from the horizontal (mode 1) and vertical (mode 0) and DC mode (mode 2) since these modes occur in the high probability. Then in the second step, we select the neighboring 22.5 degree modes (mode 5 and 7 for vertical direction, or mode 6 and 8 for horizontal direction) based on the smaller one of horizontal or vertical mode. In the third step, we refine the search further by considering the remaining neighbor mode (mode 3 or 4). We compare the best mode from the step 1 and 2 and the neighboring mode of the best one from step 2 (mode 3 or 4), and choose the best one as our final decision.

In the algorithm, three modes are initially compared in step1. In step 2, two of neighboring modes are examined to determine the refined direction. And in the last step, cost of the refine mode is calculated, and the one of the three modes are compared to make the final decision. So there are constant six modes to be examined by this algorithm.

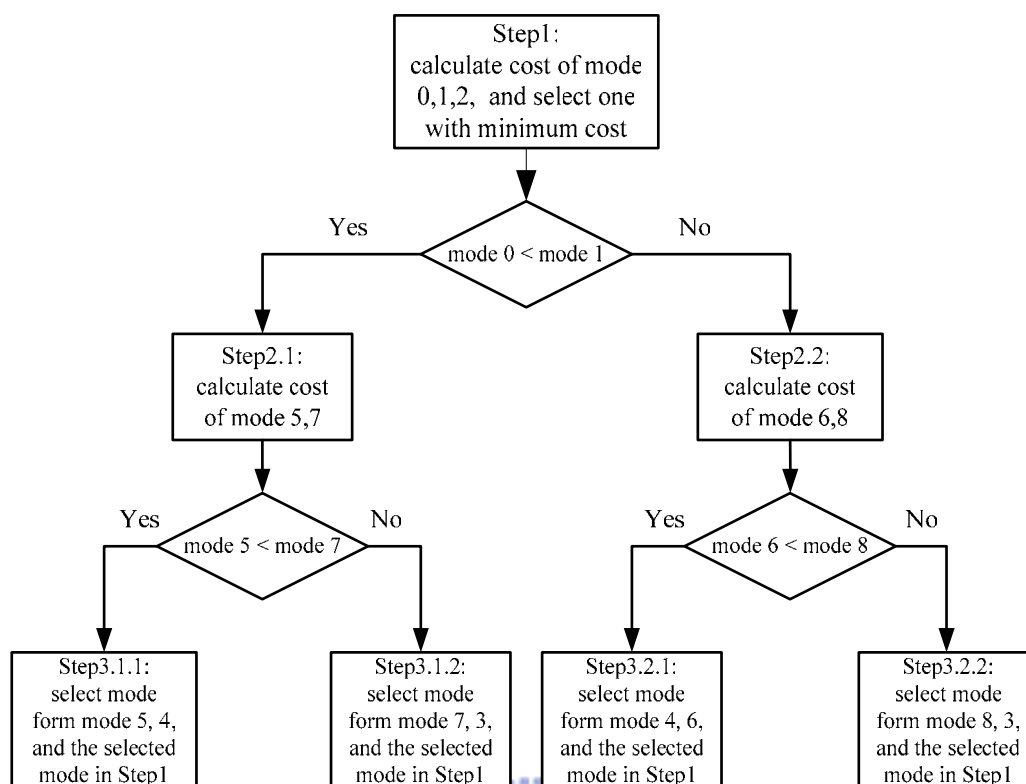


Fig. 25. Flow chart of three step intra algorithm

#### 4.4. Simulation Result and Discussion

The proposed three step algorithm and the full search are simulated on five CIF sequences, mobile and calendar, foreman, Stefan, news, and coastguard. For each sequence, 300 frames are encoded with intra frame coding. We simulate these sequences with 5 different fixed QP values, from 12 to 44 as shown in Table 5 to Table 8. RD-curve is shown in the Fig. 7 to Fig. 11.

From the result, we can find that bit-rate is increased about 1% with almost the same PSNR. We can also find bit-rate increase is step up when QP is from low to high. But when QP is high, the bit-rate increase is reduced. The phenomenon may relate to the Intra16 mode, an intra prediction mode for 16x16 blocks. In the high QP case, the opportunity to select Intra16 will also increase since Intra16 mode decision is also using full search algorithm.

For high motion and low motion test sequence, the result in bit-rate increase is almost the same. It is because that picture is intra coded without using information of other frames. The comparison with [16] is shown in Table 5 to Table 8 with four different QP values, from 28 to 40. The proposed algorithm outperforms the previous approach.

Table 5. QP = 28, Comparison results

Sequence	CHG	CHG	CHG	CHG	CHG	CHG
	BIT	BIT	PSNR	PSNR	T_I	T_AVG
	(%)	(%)	(dB)	(dB)	(%)	(%)
	[16]	Ours	[16]	Ours	Ours	Ours
Container	1.80	0.79	0.039	-0.01	-32.34	-16.03
News	2.56	1.09	0.045	-0.02	-31.05	-15.92
Paris	1.60	0.93	0.043	-0.01	-31.91	-16.62
Tempete	1.58	0.79	0.091	-0.01	-31.33	-16.22

Table 6. QP = 32, Comparison results

Sequence	CHG	CHG	CHG	CHG	CHG	CHG
	BIT	BIT	PSNR	PSNR	T_I	T_AVG
	(%)	(%)	(dB)	(dB)	(%)	(%)
	[16]	Ours	[16]	Ours	Ours	Ours
Container	2.64	1.02	0.044	-0.01	-32.94	-16.46
News	3.09	1.30	0.031	-0.02	-31.11	-15.50
Paris	2.43	1.16	0.032	-0.01	-30.29	-15.54
Tempete	2.32	0.96	0.065	-0.01	-31.43	-16.25

Table 7. QP = 36, Comparison results

Sequence	CHG	CHG	CHG	CHG	CHG	CHG
	BIT	BIT	PSNR	PSNR	T_I	T_AVG
	(%)	(%)	(dB)	(dB)	(%)	(%)
	[16]	Ours	[16]	Ours	Ours	Ours
Container	4.06	1.21	0.005	-0.02	-32.82	-16.46
News	4.26	1.20	0.000	-0.02	-31.41	-15.29
Paris	3.25	1.21	0.013	-0.01	-30.45	-15.46
Tempete	3.11	1.00	0.051	-0.02	-31.23	-16.16

Table 8. QP = 40, Comparison results

Sequence	CHG	CHG	CHG	CHG	CHG	CHG
	BIT	BIT	PSNR	PSNR	T_I	T_AVG
	(%)	(%)	(dB)	(dB)	(%)	(%)
	[16]	Ours	[16]	Ours	Ours	Ours
Container	5.18	1.00	0.001	-0.03	-32.75	-16.21
News	5.31	1.38	0.006	-0.03	-31.64	-15.16
Paris	4.91	1.58	0.003	-0.04	-30.39	-15.23
Tempete	3.67	0.78	0.024	-0.03	-31.17	-16.09

CHG BIT: change in bit-rate

CHG PSNR: change in PSNR

CHG T\_I: change in intra encoding time

CHG T\_AVG: change in average encoding time



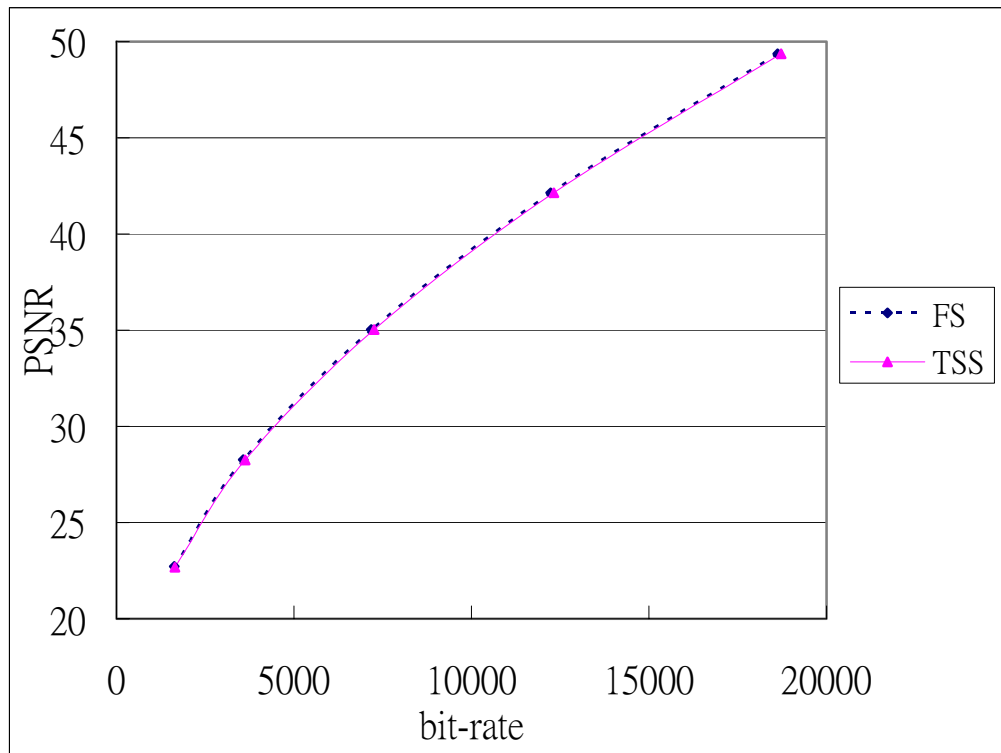


Fig. 26. RD-curve of mobile & calendar

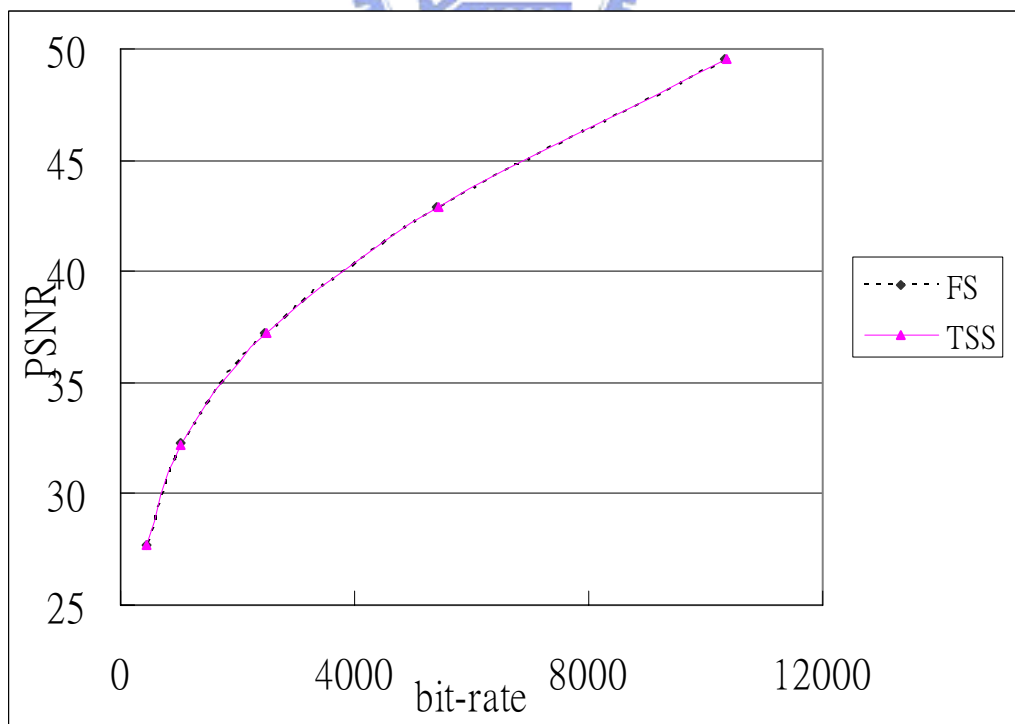


Fig. 27. RD-curve of foreman

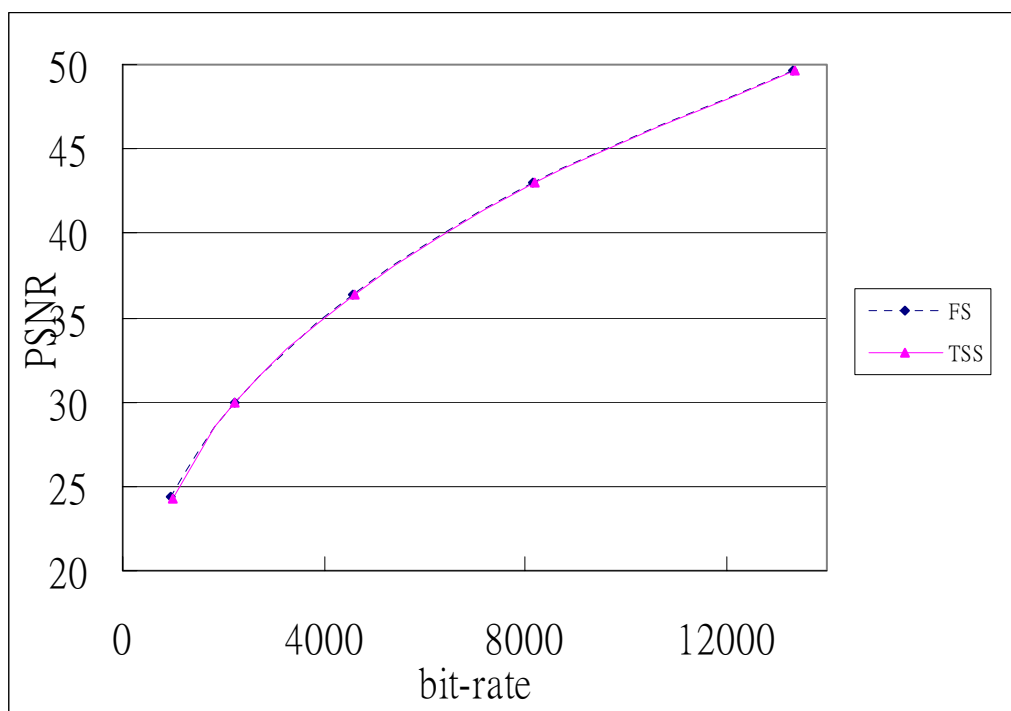


Fig. 28. RD-curve of Stefan

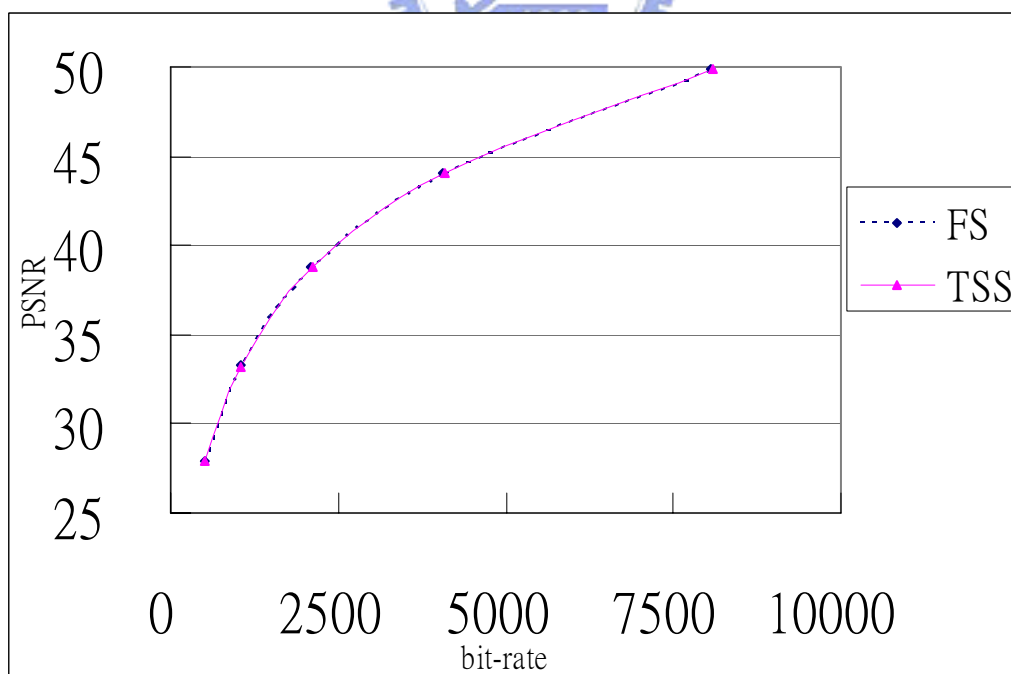


Fig. 29. RD-cure of news

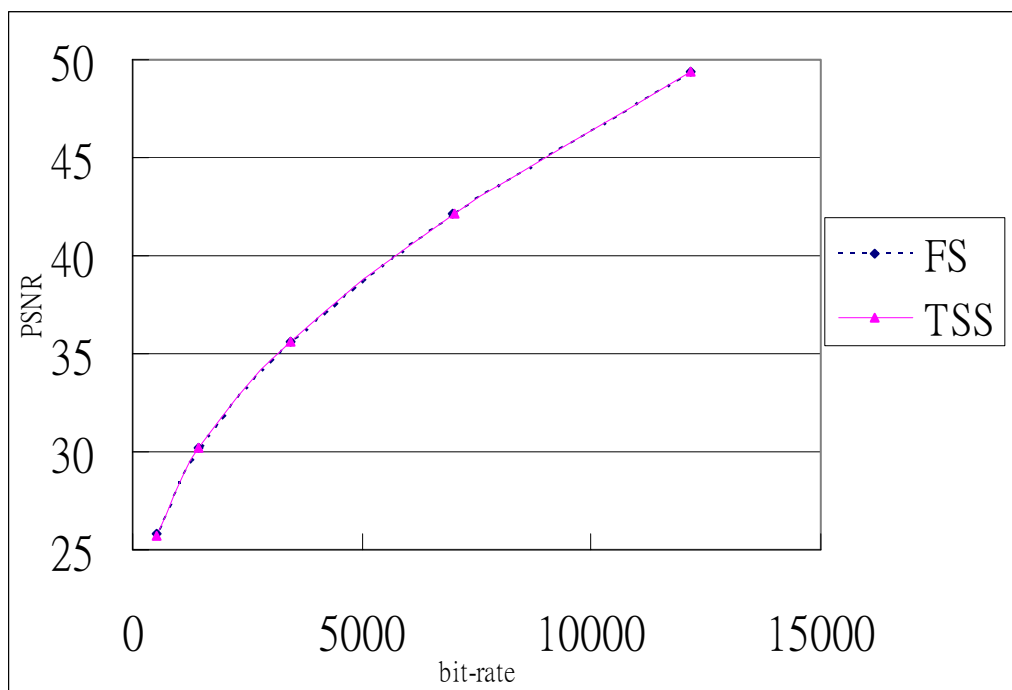


Fig. 30. RD-curve of coastguard

#### 4.5. Summary of Proposed Intra Prediction Algorithm

We propose a three step intra prediction mode selection algorithm. Computation reduction is achieved by examining only six of total intra prediction modes. Simulation results suggest that three step algorithm can achieve similar PSNR as full search and only about 1% of increase on bit-rate.

## Chapter 5 Architecture Design for H.264/AVC

### Intra Coding

H.264/AVC is regarded as the next generation video compression standard. Though original standard targeted to video applications, the high compression performance of intra-only coding also makes it suitable for still image coding, which is competitive with JPEG2000 [17].

In this chapter, an HDTV size H.264/AVC intra encoder chip for digital camera and digital video applications is presented. The chip reduces the gate count by saving the costly plane mode and enhances the video quality with the improved cost function. With careful scheduling and high performance function unit, the developed chip can easily support 29.46M pixels/s still image encoding and real-time moving picture intra coding of HDTV 720p@30fps video application when clocked at 117.28MHz under 0.18um CMOS process.

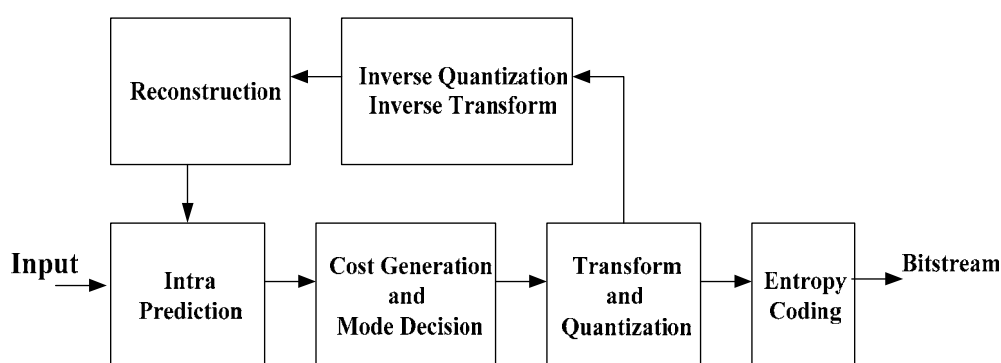


Fig. 31. Flow of H.264/AVC intra coding

### 5.1. Fundamental of H.264/AVC Intra Coding

The Intra coding flow of H.264/AVC is shown in

Fig. 31. This macroblock data will be predicted from one of nine kinds of 4x4 luma prediction modes, four kinds of 16x16 luma prediction mode, and four kinds of 8x8 chroma prediction mode. Then the prediction mode with the minimum cost value is selected as the best mode. The residuals after the prediction are further processed by transform, Q/IQ, inverse transform, and reconstructed as reference of next macroblock. The coefficients after quantization and mode information are encoded by entropy coding, CAVLC and UVLC.

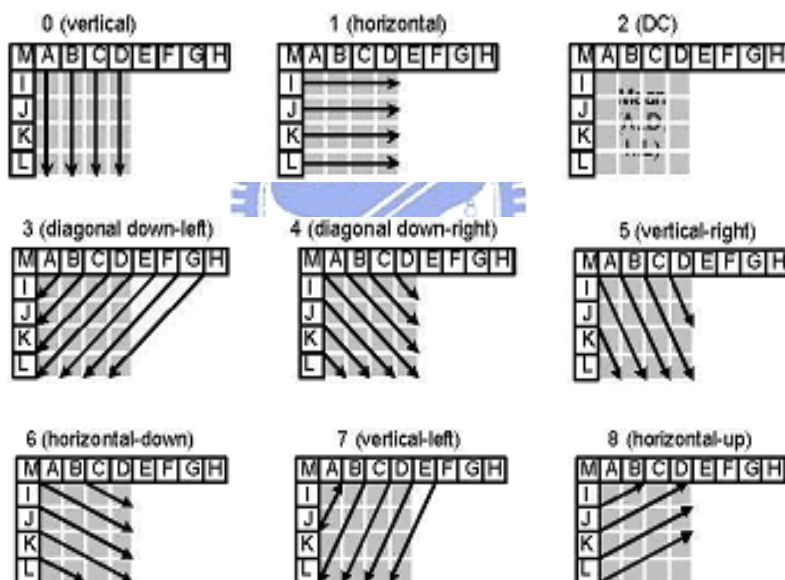


Fig. 32. Modes of Intra4x4

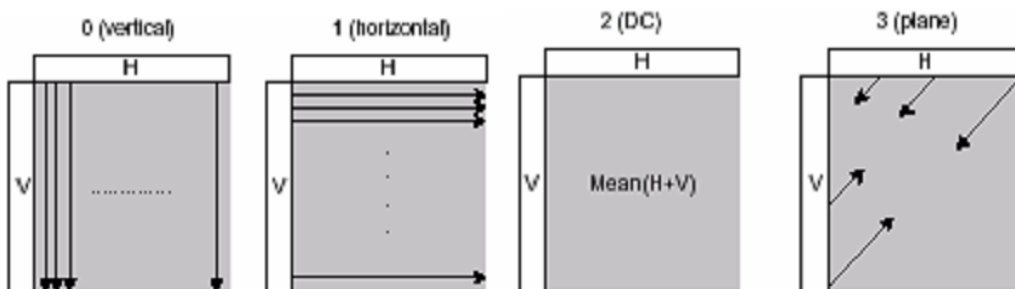


Fig. 33. Modes of Intra16x16.

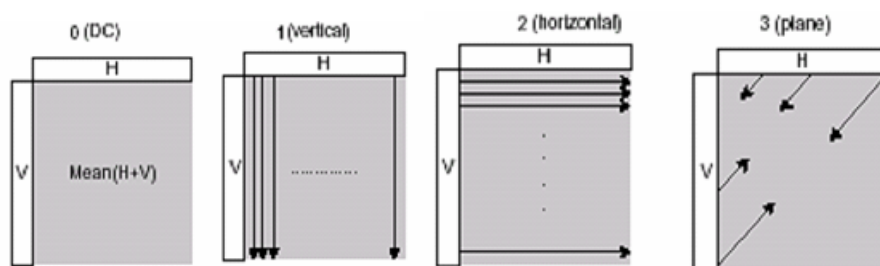


Fig. 34. Modes of chroma8x8.

### 5.1.1. Intra Prediction Mode

There are three classes of intra prediction modes. They are Intra4x4, Intra16x16 for luma sample prediction, and chroma8x8 for chroma samples prediction. Different from AC/DC prediction of MPEG-4, H.264/AVC use directional spatial information of neighbor already coded blocks to predict current sample values. Fig. 32 shows the modes of Intra4x4. Eight directional modes and one DC prediction are adopted. Fig. 33 shows the modes of Intra16x16 used for smooth texture. Intra4x4 is more suitable for high quality application while intra16x16 is suitable for low bitrate application. Fig. 34 shows the mode of chroma8x8. The mode of chroma8x8 is the same as Intra16x16 only with different mode number.

## 5.2. Hardware Oriented Algorithm Modification

### 5.2.1. Proposed Mode Decision Method

In the intra encoding flow, the mode decision method is the most important part to determine the coding performance. Two mode decision methods are used in the reference software. One is basic mode decision method and the other is rd-optimization (RDO) mode decision method.

Basic mode decision method calculates cost using table look up mode cost

and sum of absolute transform difference (SATD). RDO mode decision method use weighted sum of actual encoded bitrate and reconstructed samples to generate distortion. Though RDO mode decision method achieves the best performance, it is also computational intensive and thus is not suitable for high performance or real-time encoder implementation. Therefore, our intra encoder adopts the basic method to implement the mode decision stage as shown below

Basic cost generation function :

$$\text{Cost} = \text{Cost\_of\_Mode} + \text{SATD}$$

In the reference software, SATD is calculated by applying 4x4 discrete Hadamard transform (DHT) to the residuals of prediction modes due to its simplicity. However, since the residuals are processed by 4x4 discrete cosine transform (DCT) in the encoding flow, a 4x4 DCT transform for SATD will generate better results than DHT does, which has the side benefit to avoid computing the 4x4 DCT again.

However, 4x4 DCT in H.264/AVC is divided into two parts, 4x4 integer transform and scalar multiplication factors (the one with factors a, b) that are merged into the quantization stage, as shown in Fig. 35. The reference software adopts DHT simply for its simplicity to approximate the 4x4 integer transform. A better way for SATD calculation is to approximate the 4x4 DCT, but this should have low computational complexity as DHT does.

$$\begin{pmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \\ \mathbf{X} \\ \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \end{pmatrix} \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

Fig. 35. 4X4 DCT transform of H.264/AVC

QP	Positions (0,0),(2,0),(2,2),(0,2)	Positions (1,1),(1,3),(3,1),(3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

Fig. 36. quant\_coef table of quantization

QP	Positions (0,0),(2,0),(2,2),(0,2)	Positions (1,1),(1,3),(3,1),(3,3)	Other positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

Fig. 37. dequant\_coef table of inverse quantization.

First, we look at the equation of quantization and inverse quantization

Quantization

$$- L = (\text{abs}(M) * \text{quant\_coef} + \text{qp\_const}) \gg \text{q\_bits}$$

Inverse quantization

$$- L * \text{dequant\_coef} \ll \text{qp\_per}$$

qp\_per, q\_bits and qp\_const are derived from quantization parameter

Quantization is calculated by using a table look up constant multiplication and an offset derived from quantization parameter. Inverse quantization is calculated only by a table loop up constant multiplication. We use the quantization factors, quant\_coef, shown in Fig. 36 or inverse quantization factor, dequant\_coef, shown in Fig. 37 to derive the scaling factors.

$$- 1/\text{quant\_coef}: [0][0]:[0][1]:[1][1] \sim 30:19:12$$

$$- 1/\text{dequant\_coef}: [0][0]:[0][1]:[1][1] \sim 32:25:20$$



$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \mathbf{X} \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{pmatrix} \otimes \begin{bmatrix} 32 & 25 & 32 & 25 \\ 25 & 20 & 25 & 20 \\ 32 & 25 & 32 & 25 \\ 25 & 20 & 25 & 20 \end{bmatrix} \gg 5$$

Fig. 38. Modified SATD calculation method

Fig. 38 shows our modified method of SATD calculation. In our simulation, the scalar factors derived from inverse quantization is better than factors from quantization. The reason is that quantization process is also affected by an offset  $qp\_const$ . The result of modified mode decision method is better than the reference software.

### 5.2.2. Intra Prediction Mode

In H.264/AVC Intra coding, intra prediction and mode decision are the two computation extensive components. All prediction modes are examined to find the best mode. Parallel architecture are demanded to accelerate these components. After analyzing the type of intra prediction modes, we can separate the modes into four types as shown in Fig. 39. In the bypass type, prediction samples are the same as boundary pixels. In the linear types, prediction samples are linear interpolation derived from boundary pixels. In the average type, prediction samples are average of all boundary pixels. In the plane type, prediction samples are approximation of bilinear transform with only integer arithmetic as shown in Fig. 40. The equation of Plane mode is more complex than other modes and is hard to reuse with other mode.

However, by simulation we found that intra prediction with plane prediction mode only reduces about 1% of bit-rate than that without plane mode. This 1% of bit-rate difference can be easily compensated by the enhanced cost function and achieves almost the same result with the basic method in reference software

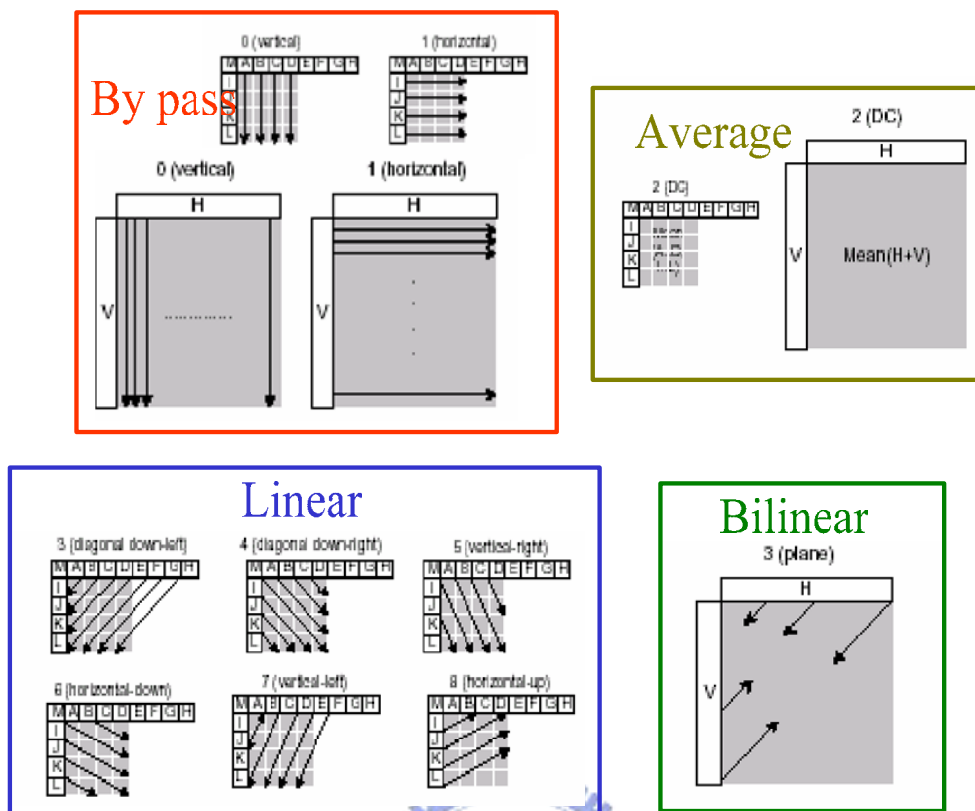
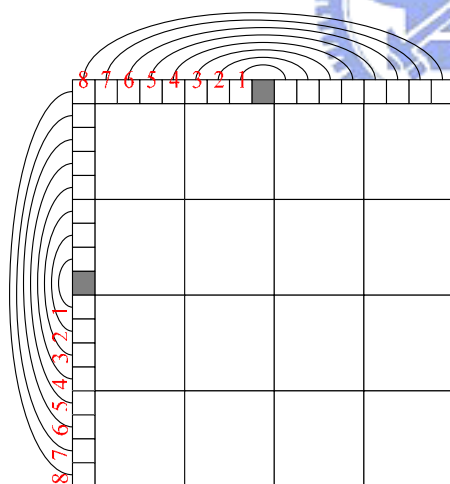


Fig. 39 Four types of intra prediction modes



**Luma 16x16**

$$h = \sum_{i=1}^8 x[p(7+i,-1) - p(7-i,-1)]$$

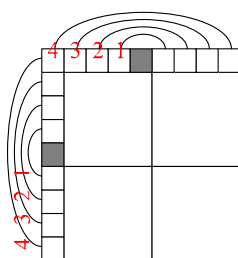
$$v = \sum_{i=1}^8 y[p(-1,7+i) - p(-1,7-i)]$$

$$a = 16 * [p(-1,15) + p(15,-1)]$$

$$b = (5h + 32) \gg 6$$

$$c = (5v + 32) \gg 6$$

$$\text{Pred} = [a + b(x-7) + c(y-7) + 16] \gg 5$$



**Chroma 8x8**

$$h = \sum_{i=1}^4 x[p(4+i,-1) - p(4-i,-1)]$$

$$v = \sum_{i=1}^4 y[p(-1,4+i) - p(-1,4-i)]$$

$$a = 16 * [p(-1,7) + p(7,-1)]$$

$$b = (17h + 16) \gg 5$$

$$c = (17v + 16) \gg 5$$

$$\text{Pred} = [a + b(x-3) + c(y-3) + 16] \gg 5$$

Fig. 40. Equations of plane mode prediction

The simulation result is shown from Fig. 41 to Fig. 48. Thus, we decide to implement the intra coding without plane prediction mode based on the cost and performance trade-off.

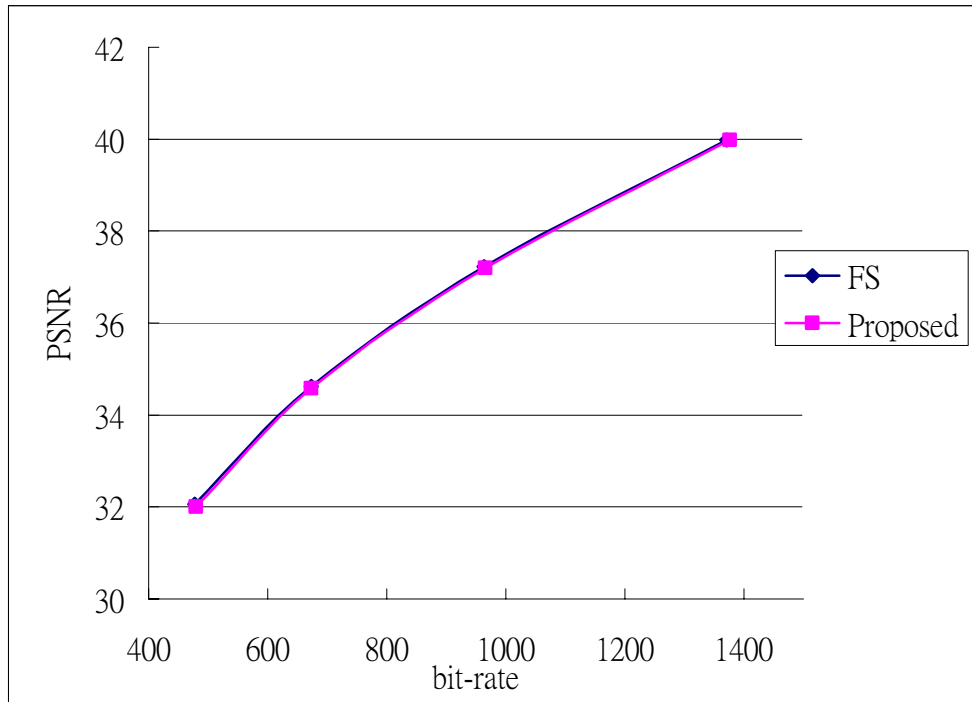


Fig. 41. RD curve of Akiyo

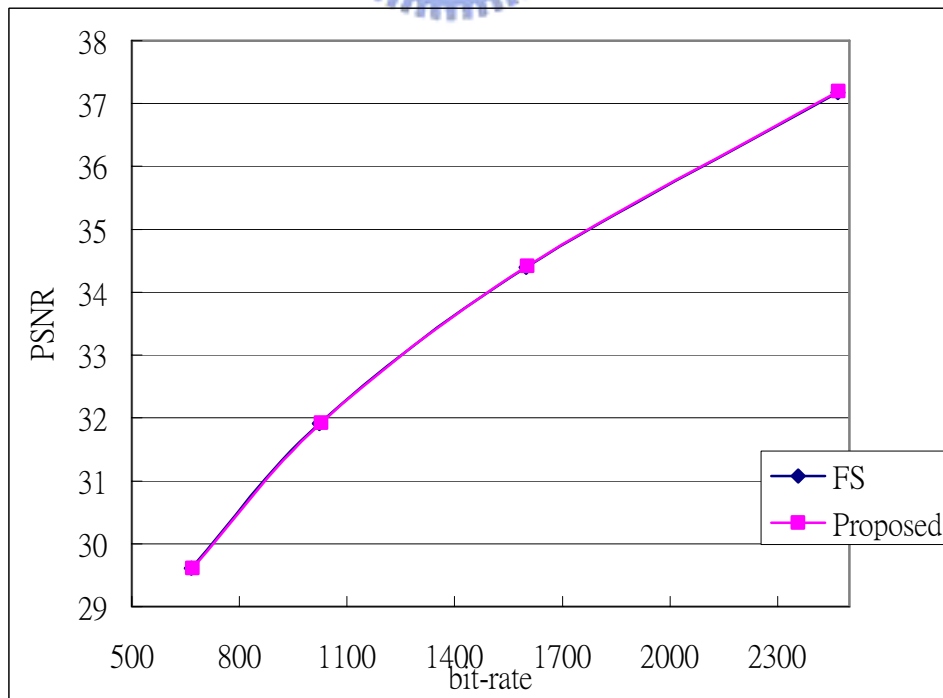


Fig. 42. RD curve of Foreman

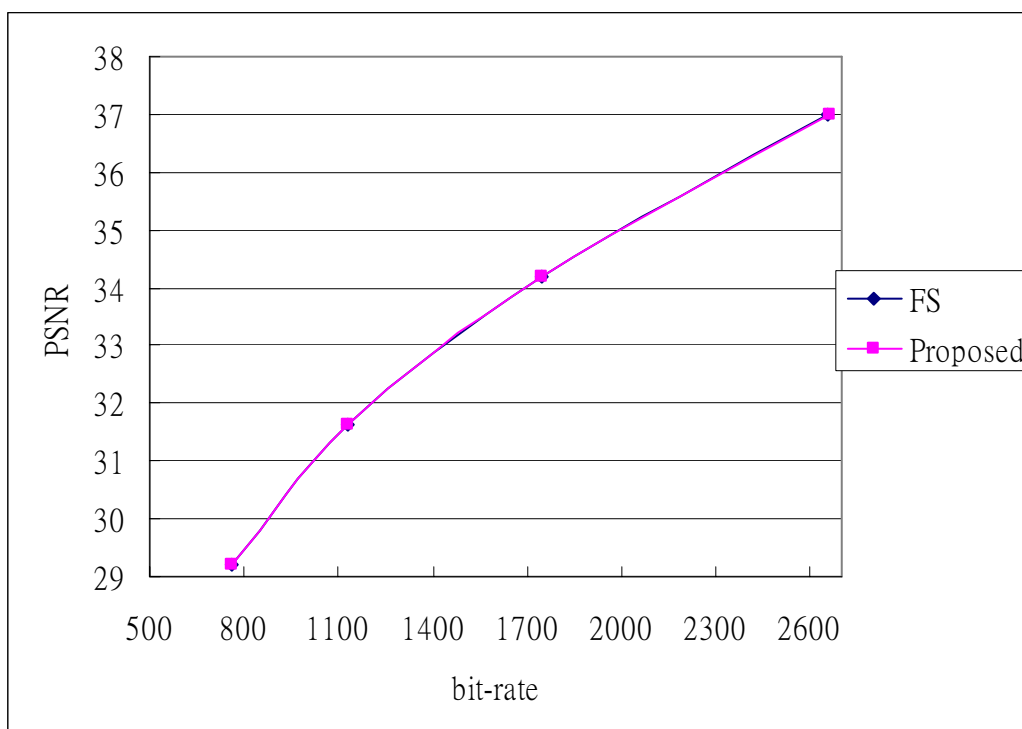


Fig. 43. RD curve of container

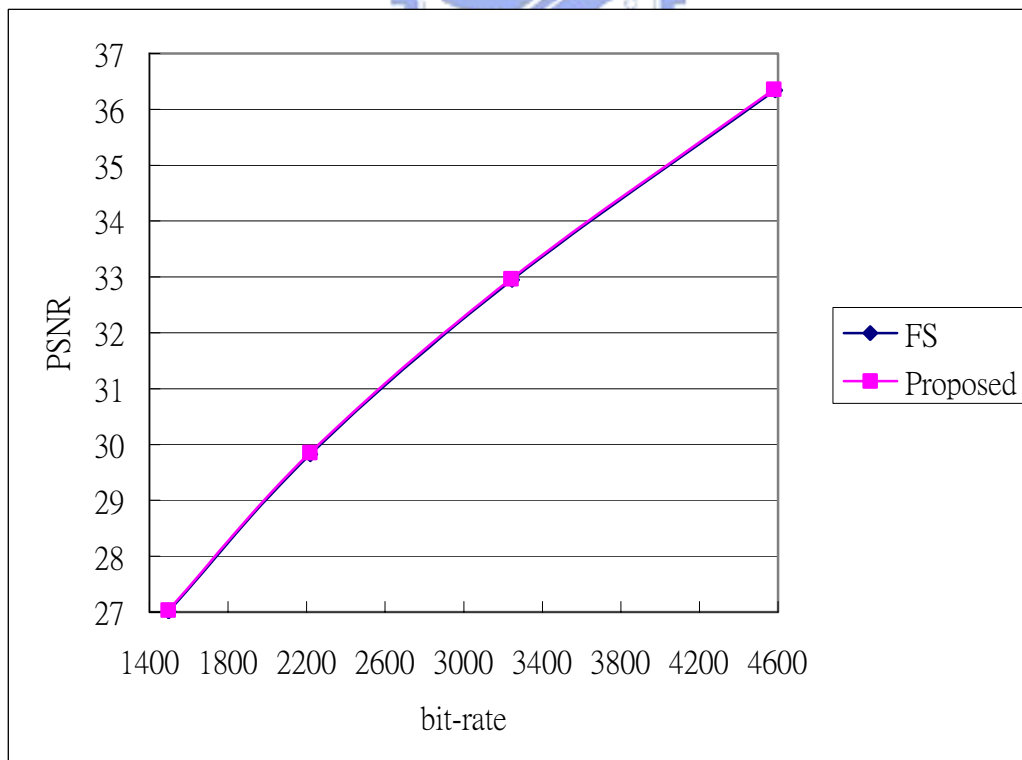


Fig. 44. RD curve of stefan

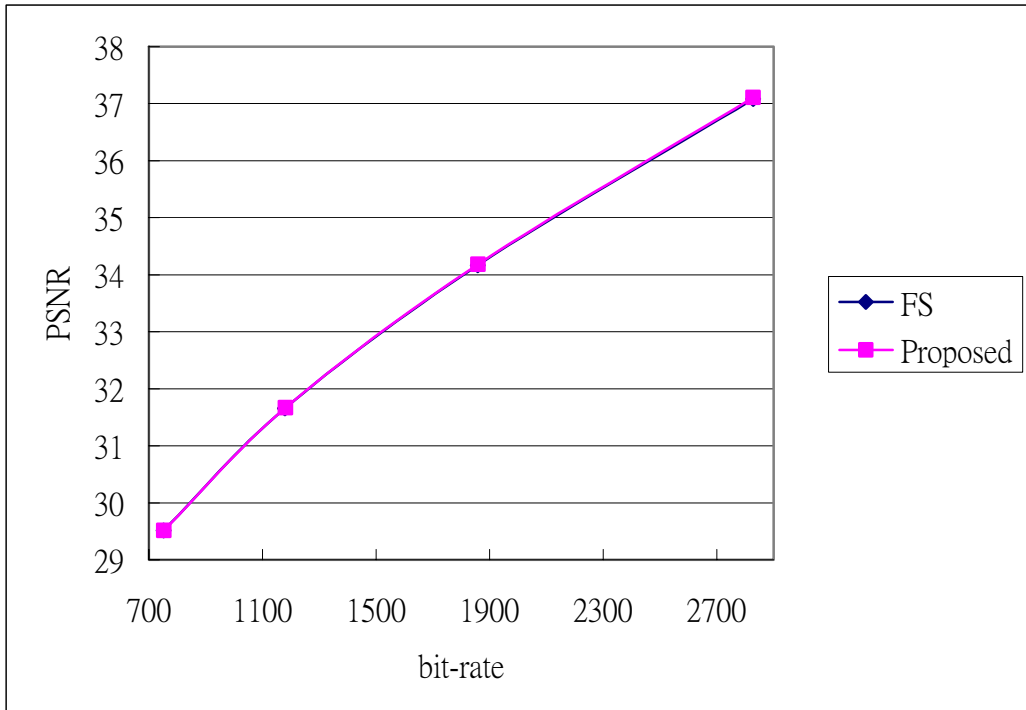


Fig. 45. RD curve of football

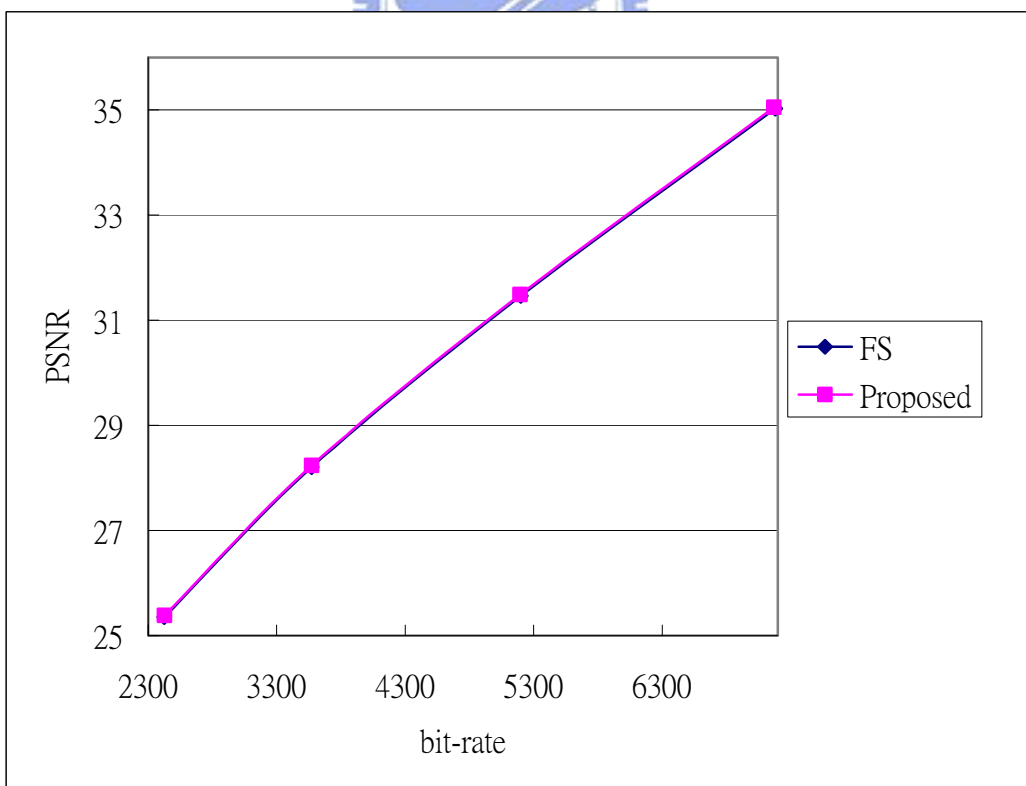


Fig. 46. RD curve of mobile and calendar

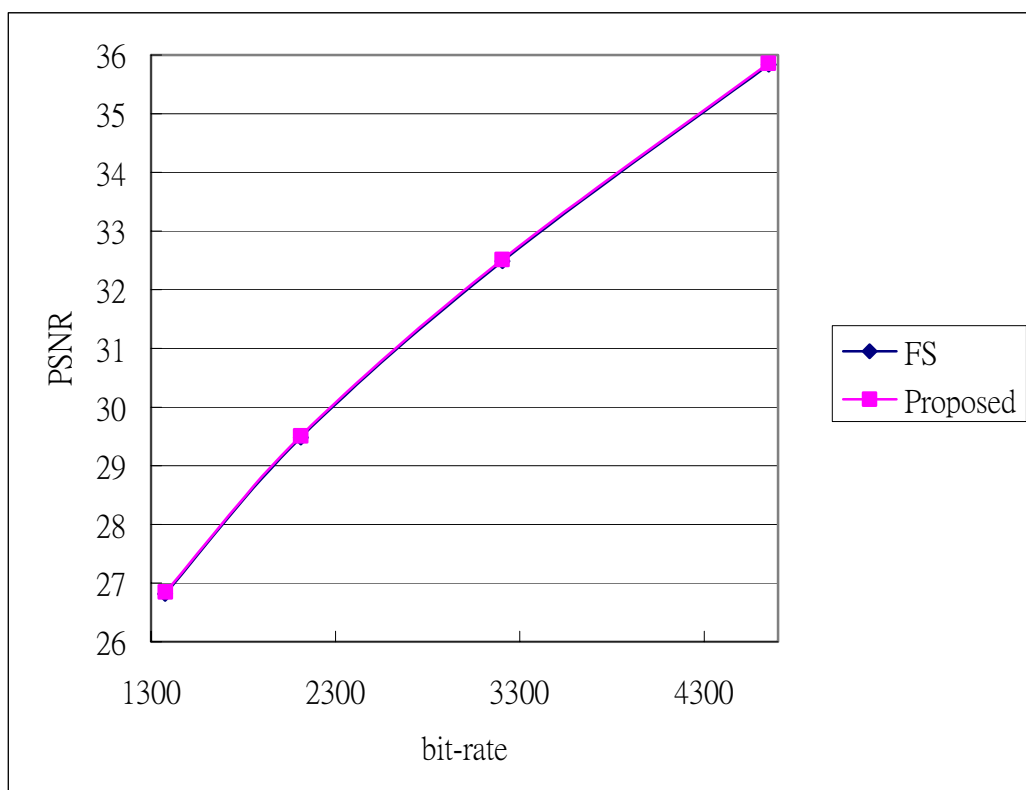


Fig. 47. RD curve of tempete

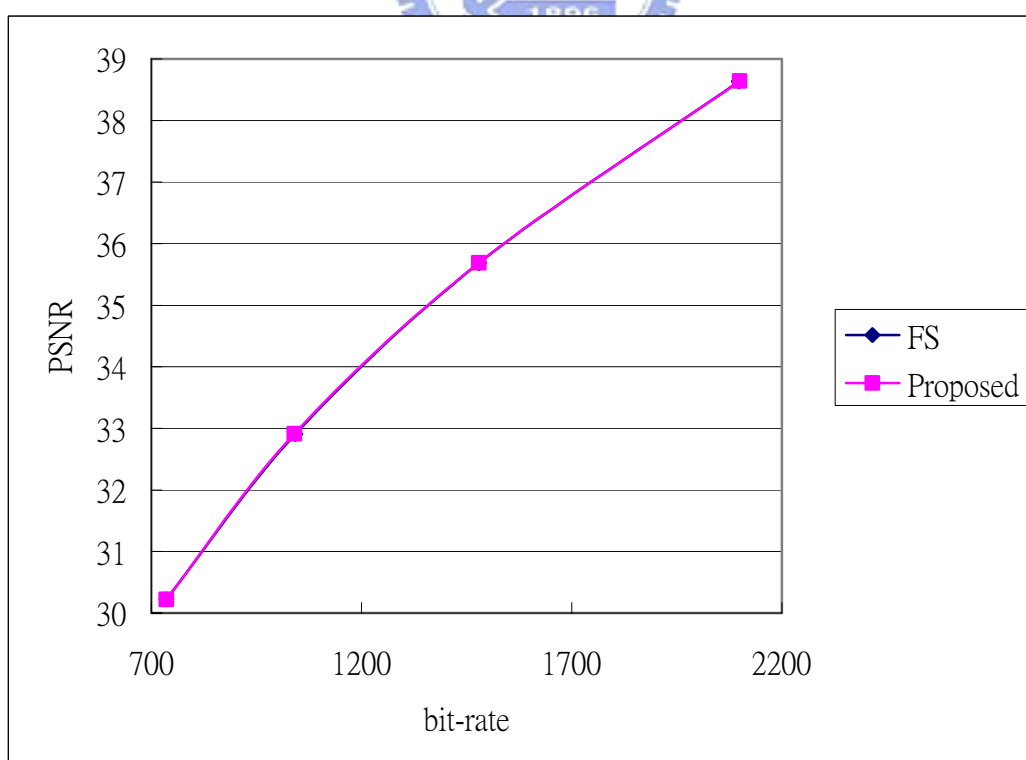


Fig. 48. RD curve of news

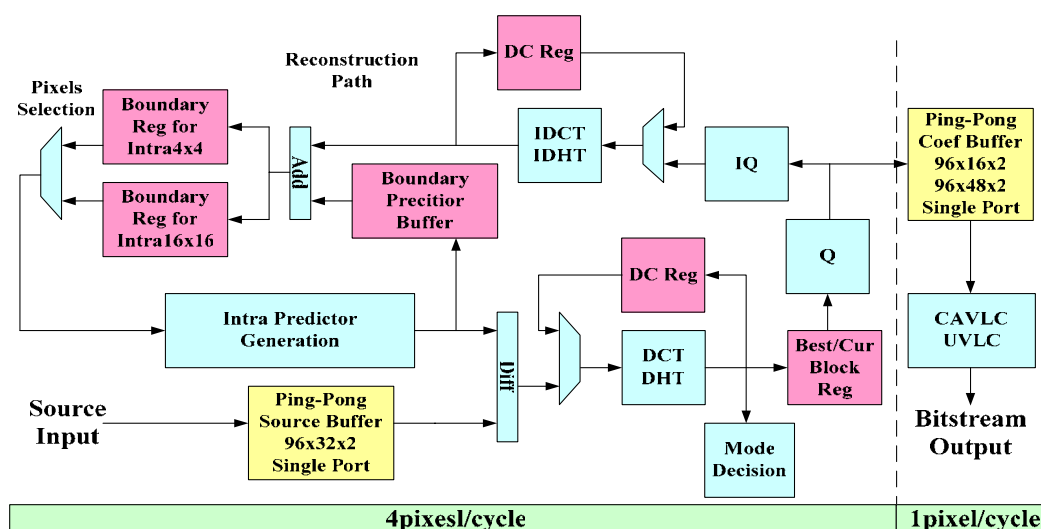


Fig. 49 Architecture of Intra Coding

## 5.3. Architecture Design of H.264/AVC Intra Coding

### 5.3.1. System Architecture Design

Fig. 49 shows the intra encoding architecture, which is directly corresponding to the coding flow shown in Fig. 31. The architecture consists of the intra prediction unit, transform unit, quantization unit and CAVLC unit. First, the intra prediction unit will generate the prediction value for the current block. Then for each possible mode, the residual pixels after prediction are transformed by 4x4 integer transform or DHT (DC value of Intra16x16 or Chroma8x8). These transform coefficients are further used to compute the cost function to determine the best by the proposed cost function. The intra4x4 block with lower cost is preserved in the buffer. After best intra4x4 block is obtained, it will go through the reconstruction path to generate the required boundary samples for the next 4x4 block. The data after quantization and mode information will be coded by CAVLC and UVLC, respectively.

In the intra encoder implementation, the major bottleneck is the feedback loop in the reconstruction path since the next 4x4 block cannot start its computation until its boundary samples are reconstructed from previous blocks. Thus, three scheduling techniques are proposed to accelerate this data dependency problem.

1. Insertion of intra16x16 prediction: During the empty bubble cycles of intra4x4 block reconstruction, intra16x16 prediction process is inserted into these bubble cycles of intra predictor generation unit to pre-compute the Intra16 cost. Thus, the utilization of intra predictor is improved.
2. Early start of next 4x4 block prediction: before the boundary samples are available, the prediction mode using upper samples (vertical prediction mode) can be early started before other modes.
3. Intra16x16 DC value pre-computing: In the H.264/AVC standard, the sixteen DC coefficients from the Intra16x16 mode have to be transformed again by DHT. Thus, for the reconstruction, inverse DCT of other AC coefficients cannot be started before inverse DHT, and this situation will result in a macroblock size buffer to store the AC coefficient of sixteen 4x4 blocks. Using the intra16x16 prediction insertion mentioned in technique 1, the best intra16x16 DC value after DHT is pre-computed from the Q/IQ stage to the DC registers of IDCT/IDHT stage. Not only a macroblock size buffer is saved but also the overall computation cycles are reduced.



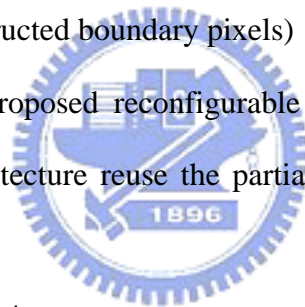
### 5.3.2. Intra Predictor Generation Unit

A reconfigurable 4 pixels parallel intra predictor generation unit is proposed. It can support nine kinds of Intra 4x4 modes, three kinds of Intra16x16 modes, and three kinds of Chroma8x8 modes. After analyzing the prediction mode, we can find that prediction samples are derived from boundary pixels using four types of arithmetic equation:

1.  $(A+B+1) \gg 1$
2.  $(A+2B+C+2) \gg 2$
3. Bypass (for Vertical, Horizontal mode)
4. DC (Intra4x4: average of 8 pixels, Intra16x16: average of 32 pixels)

(A, B and C are reconstructed boundary pixels)

Fig. 50 shows the proposed reconfigurable architecture of intra predictor generation unit. The architecture reuse the partial sum of neighbor predictor to save the adder count.



For example : Intra4x4

$$\text{Predictor1} = B+2C+D = (B+C)+(C+D)$$

$$\text{Predictor2} = A+2B+C = (A+B)+(B+C)$$

Thus, B+C can be reused to generate two predictor output

Some examples are shown in Fig. 51 to Fig. 55.

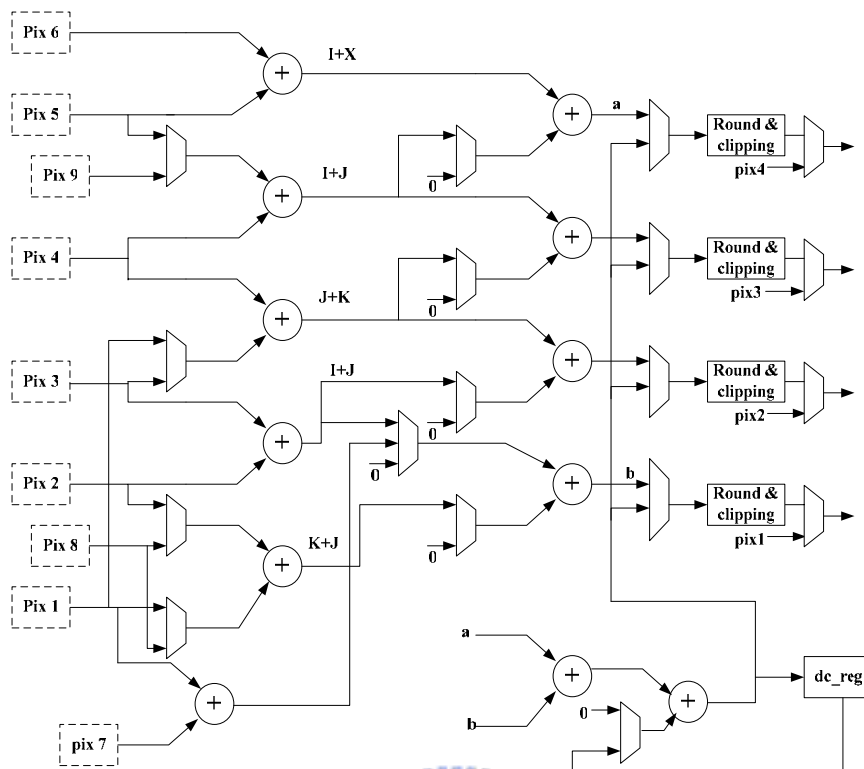


Fig. 50. Reconfigurable data path of intra predictor generation unit

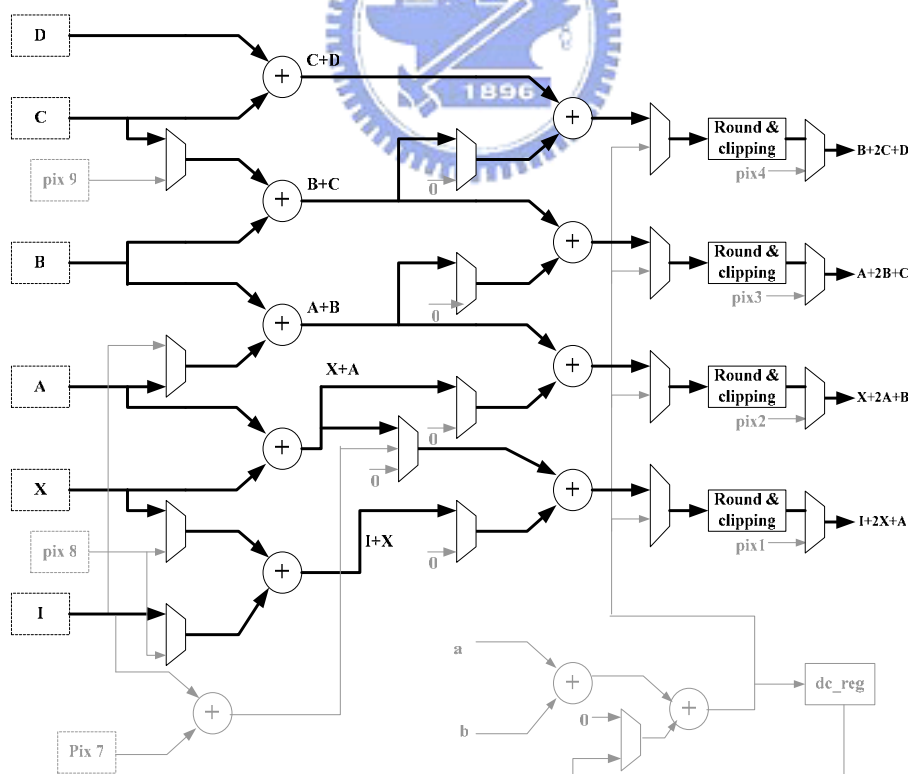


Fig. 51. Data path of diagonal down right

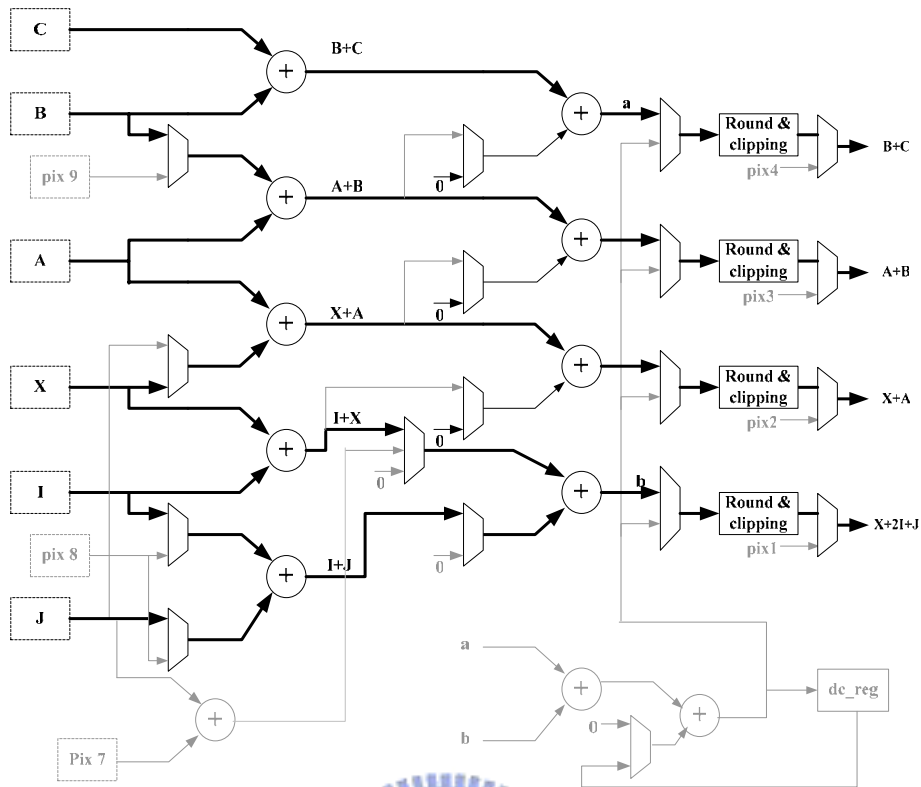


Fig. 52. Data path of vertical right

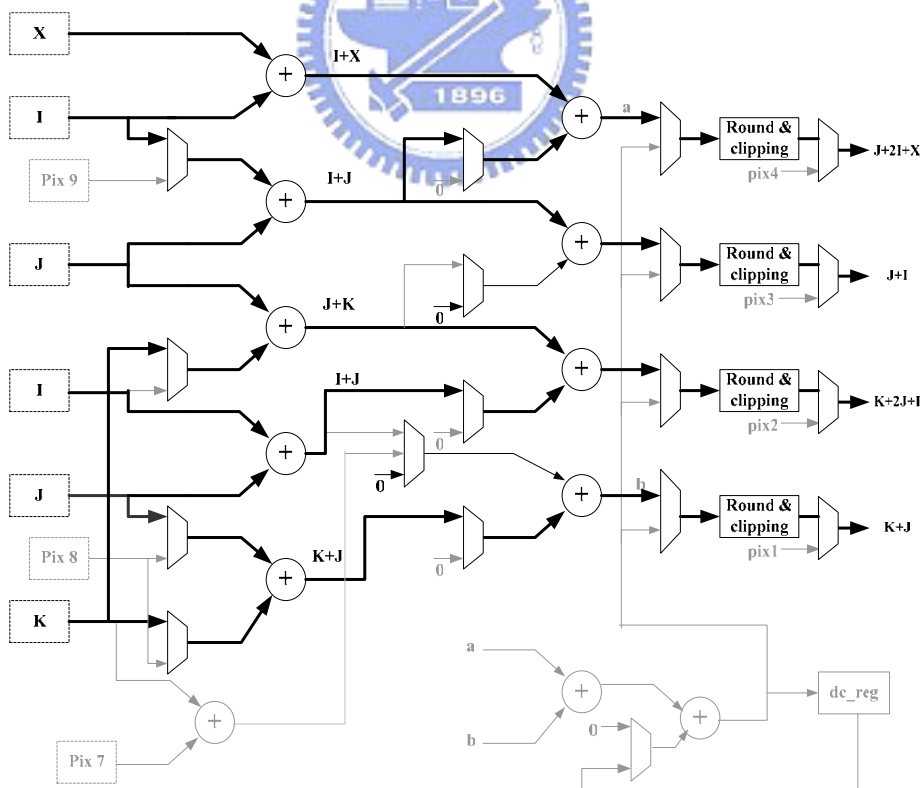


Fig. 53. Data path of horizontal down mode

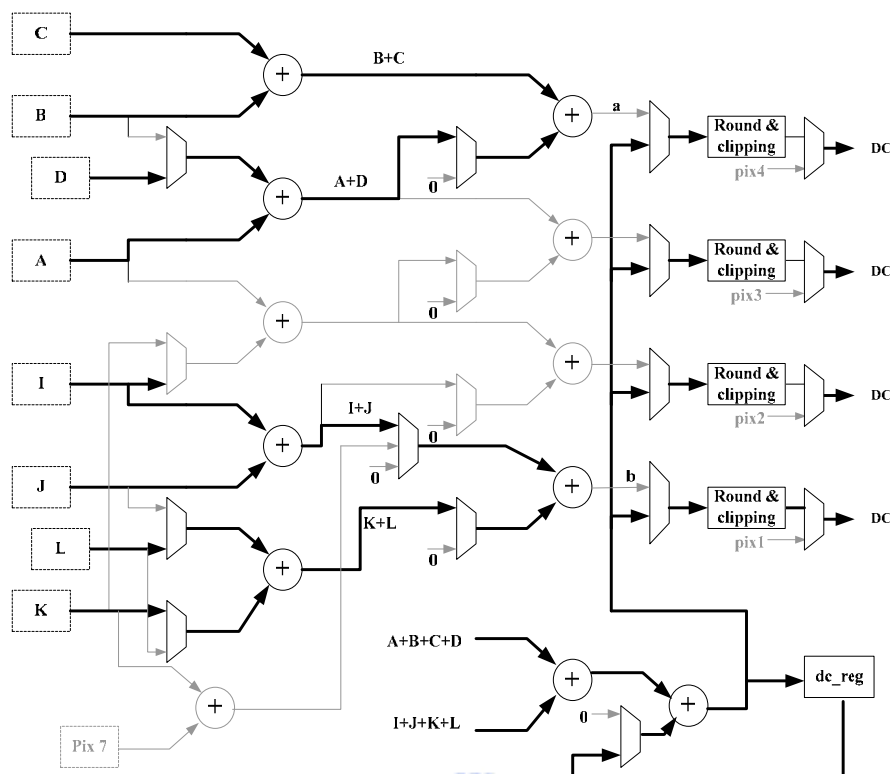


Fig. 54. Data path of DC prediction mode

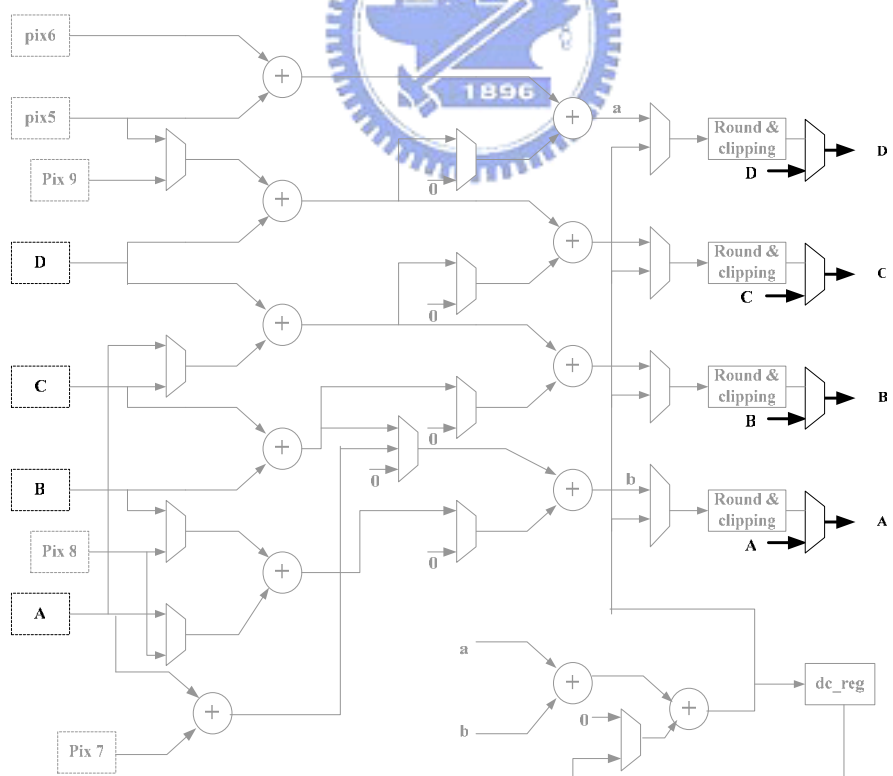


Fig. 55. Data path of horizontal mode

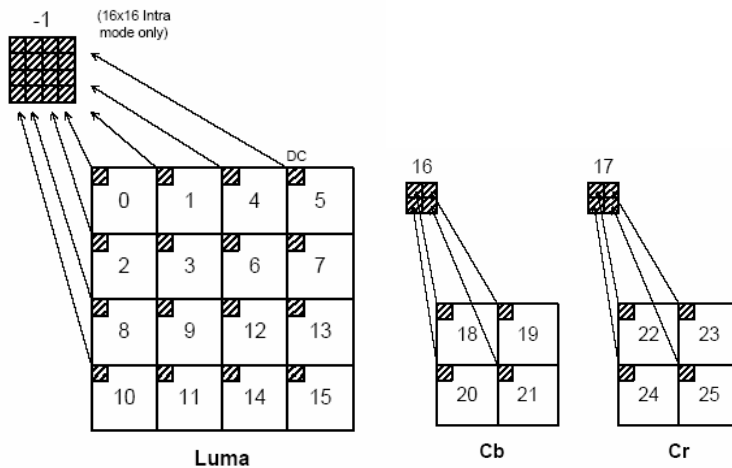


Fig. 56. Coding order of residual blocks

### 5.3.3. Transform Unit

In H.264/AVC, residual macroblock is divided in 16 4x4 luma blocks and 8 4x4 blocks as shown in Fig. 56. All the 4x4 blocks will be transformed with integer coefficient. If the intra prediction mode is Intra16x16, the DC value of 16 luma blocks will be transformed again by 4x4 discrete Hadamard transform. The 2x2 DC values of chroma blocks after DCT will also be transformed by 2x2 DHT.

Transform matrix of DCT, IDCT, and Hadamard transform is shown in Fig. 57 to Fig. 59. We can find the coefficients of the transform matrixes are even or odd symmetry at each row or column and can be implemented by add and shift. The number of addition in each 1D transform can be reduced from 16 to 8 with butterflies. Fast algorithm and its butterfly structure are shown in Fig. 60. Because two forward transforms have the same structure and will not operate at the same time in our system architecture. We can merge them together to save area. Inverse transform of DCT and DHT are merged by the same method as the forward methods. The transform unit handles uses the similar architecture as in [18]. Fig. 61 shows the hardware architecture of transform unit.

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}$$

Fig. 57. Transform matrix of 4x4 DCT transform

$$f = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

Fig. 58. Transform matrix of 4x4 IDCT transform

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Fig. 59. Transform matrix of Hadamard transform

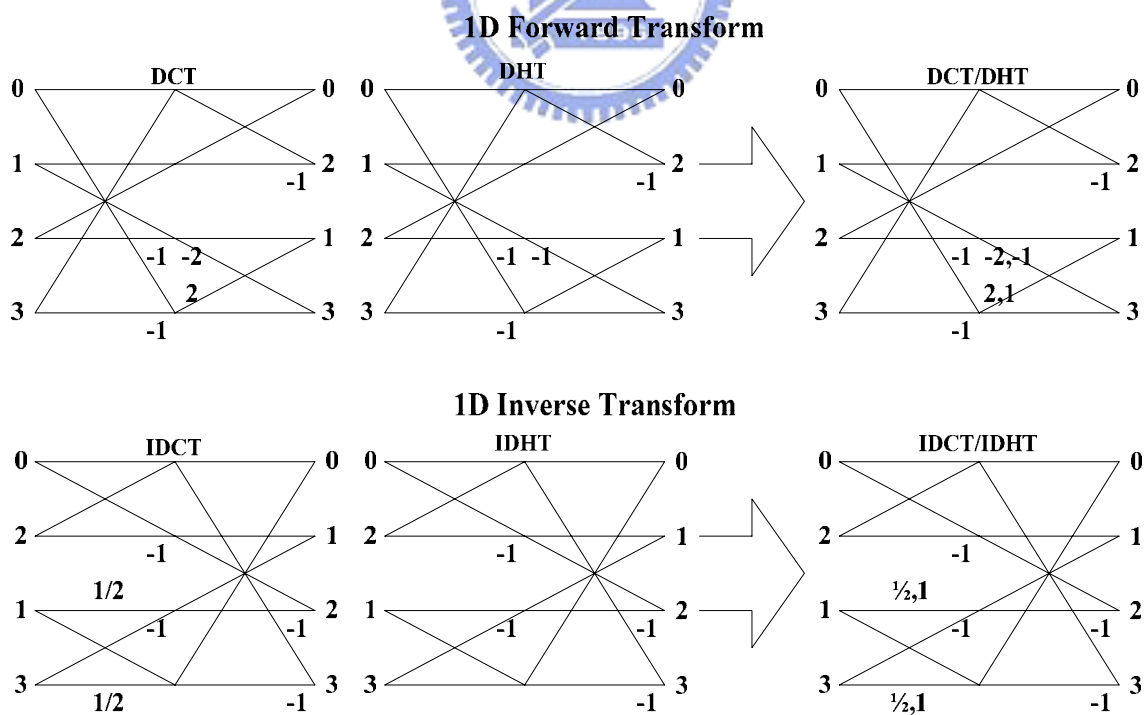


Fig. 60. Fast algorithms of 4x4 transform

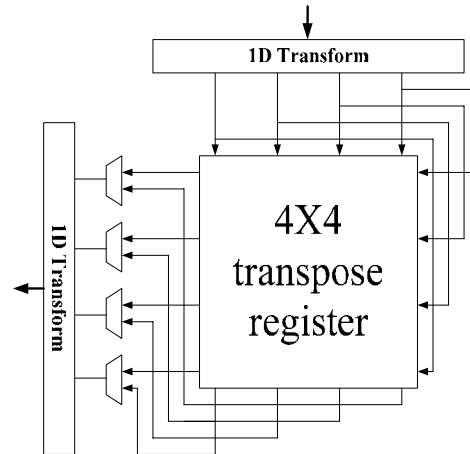


Fig. 61. Hardware architecture of transform unit

### 5.3.4. Quantization Unit

The quantization and inverse quantization unit are shown in Fig. 62. The constant value of `quant_coef`, `dequant_coef`, `qp_const`, `qp_shift`, and `qp_per` are implemented by look-up table depending on the QP values. The design also uses the data guarding technique to reduce power consumption when input value is zero.

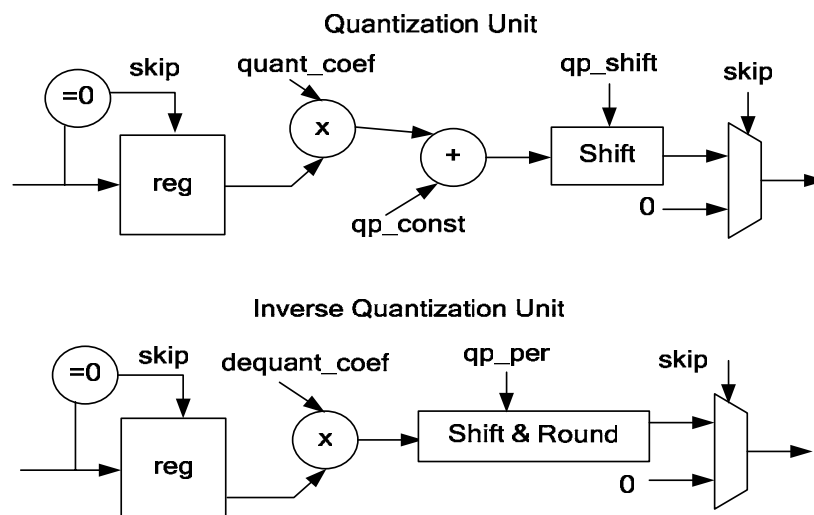


Fig. 62. Hardware architecture of quantization unit

### 5.3.5. Mode Decision Unit

Fig. 63 shows the hardware architecture of mode decision unit. The transformed coefficient is accumulated by mode decision unit. The scaling operations are implemented by shift and add. If the cost of current mode is small than best mode, the cost and mode value of best mode register will be refreshed. After processing whole macroblock, the mode with minimum cost will be selected as the best intra prediction mode.

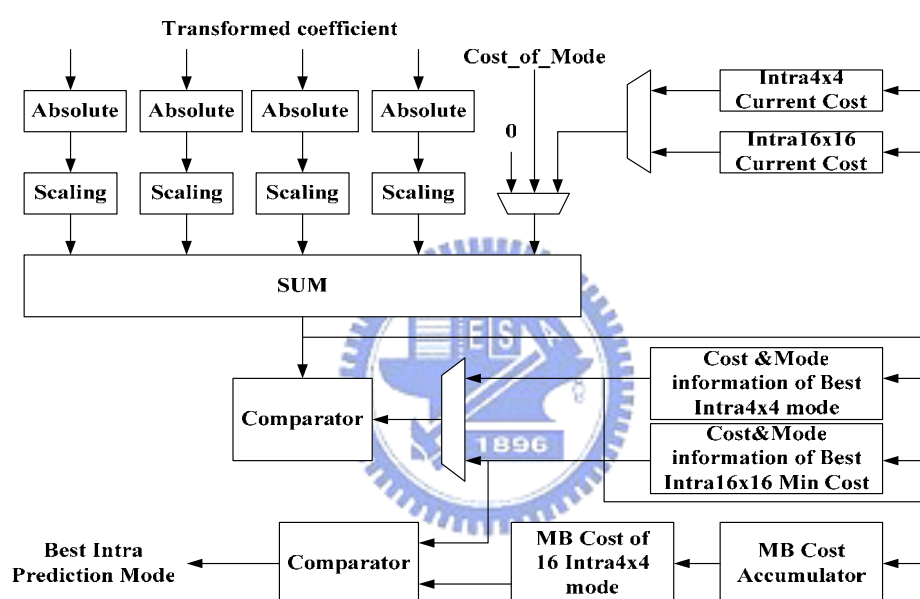


Fig. 63. Hardware architecture of mode decision unit

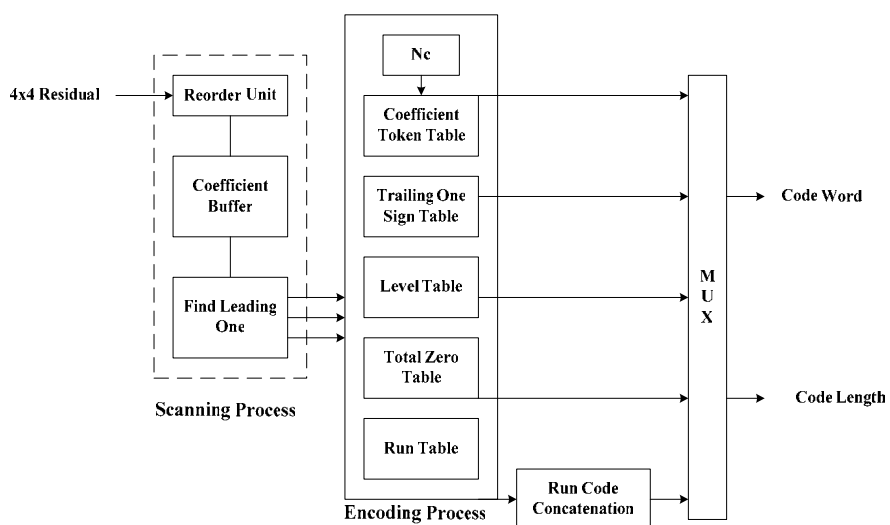


Fig. 64. CAVLC architecture



### 5.3.6. CAVLC Unit

The architecture of CAVLC is shown in Fig. 64. CAVLC encoding process can be divided into two phases, scanning phase and encoding phase. Input of CAVLC is four transformed coefficients per cycle. The scanning phase will skip the zero coefficients and only scans the nonzero one in the inverse zigzag scan order to speedup the encoding phase. Then, the data are sent to the corresponding lookup tables in parallel. These codes are buffered and concatenated to form the final bitstream.

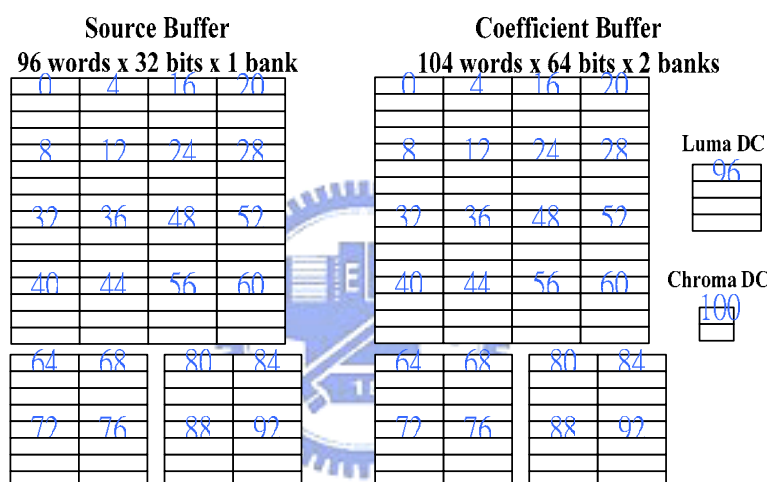


Fig. 65. Memory Organization

### 5.3.7. Memory Organization

In the proposed architecture, two components have memories. The organizations of memories are shown in Fig. 65. Source buffer stores the input data 4 pixels row by row. Coefficient Buffer is divided into two parts to facilitate DC value access in Intra16x16 mode. By using Ping-Pong architecture, data input phase and entropy coding phase can be pipelined to improve the encoding throughput.

### 5.3.8. Overall Architecture Performance

Fig. 66 shows the timing schedule of proposed intra coder. Intra16x16 prediction mode is inserted in the Intra4x4 reconstruction cycle. If Intra16x16 is selected as best prediction mode, the quantization coefficient will be recomputed again to replace the data in ping-pong buffer.

1086 cycles are spent for pipelined architecture as shown in Fig. 67. The performance of proposed architecture only needs about 117.28MHz to meet HDTV 720p (1280x720@30Hz) real-time application.

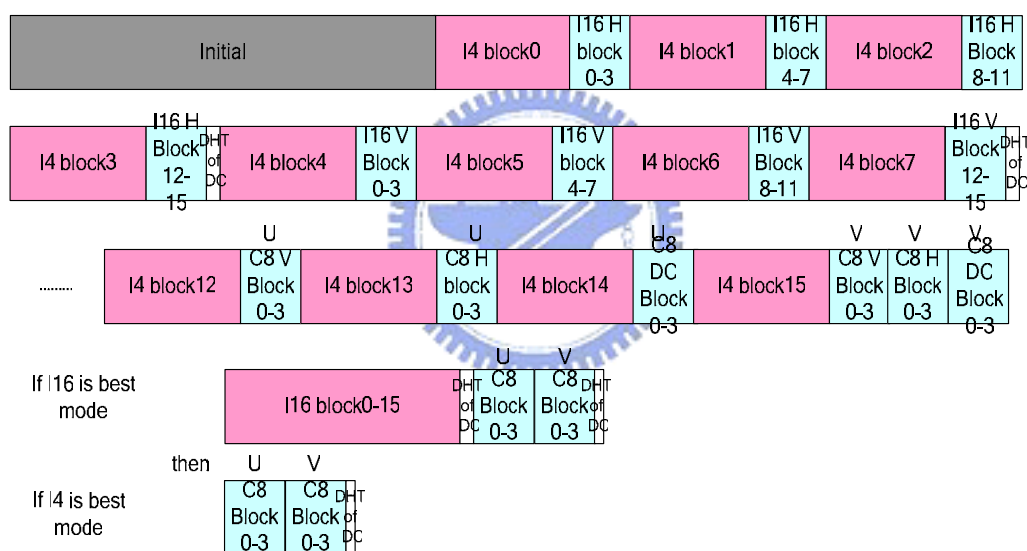


Fig. 66. Timing schedule of proposed intra coder.

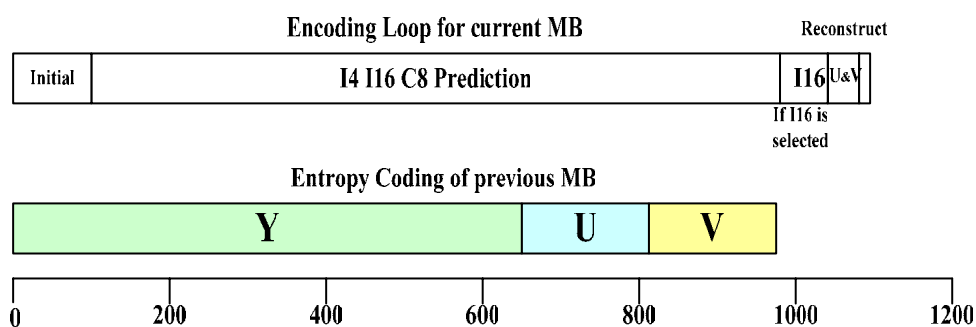


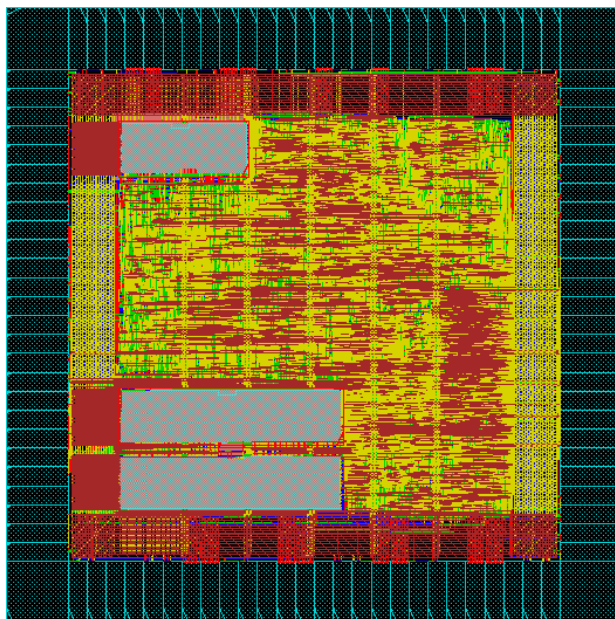
Fig. 67. Timing schedule of proposed architecture

## 5.4. Implementation Results

To evaluate the accuracy and the efficiency of the proposed architecture, the design is implemented using the UMC 0.18 $\mu$ m 1P6M CMOS technology and the cell-based design flow. The chip has an area of 2.4x2.4 mm<sup>2</sup> (pad limited) as shown in Fig. 68. The design can achieve 125 MHz at the worst-case. Thus, it can easily support 29.46M pixels/s still image encoding and real-time moving picture intra coding of HDTV 720p@30fps video application when clocked at 117.28MHz. Therefore, it is suitable for digital video or camera applications.

Table 9. List of gate count

Intra Predictor	3507
Q/IQ	22082
DCT(with DC register)	9985
IDCT(with DC register)	9836
Boundary Reconstruction Unit	15697
Cost Generation and Mode Decision Unit	10315
UVLC/CAVLC	11965
Controller	2781
Boundary Predictor Buffer	6465
Total	92633

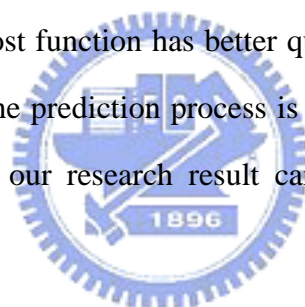


Technology:	UMC 0.18 $\mu\text{m}$ 1P6M CMOS
Voltage:	1.8 V (Core)
	3.3 V (I/O)
Die Size:	2.4x2.4 mm <sup>2</sup>
Core size:	1.28x1.28mm
SRAM: (all single port)	
Coefficient buffer	104 x 64 bits x 2 banks
Source buffer	96 x 32 bits x 1 bank

Fig. 68 Chip specification

## Chapter 6 Conclusion

In this thesis, our contribution is in three parts. The first contribution is the deblocking filter architecture that can accelerate the deblocking process. The proposed two architectures not only save the memory size but also have higher speed. The idea is to rearrange the data flow and achieve higher data reusability. The second contribution is the fast intra coding algorithm can reduce the computational complexity of intra 4x4 prediction. Six modes are required instead of nine modes in the full search method. The fast intra prediction algorithm can save 33% computational complexity with only about 1% bit-rate loss. The final contribution is the intra coding architecture can speed up the computation of intra frame coding. Proposed cost function has better quality and complex plane mode is skipped to save area. The prediction process is well scheduled to achieve high utilization. We hope that our research result can promote the convenience of human life.



## Bibliography

- [1] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification* (ITU-T Rec. H.264/ ISO/ IEC 14496-10 AVC), Mar. 2003.
- [2] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003
- [3] Information Technology - Generic Coding of Moving Picture and Associated Audio Information: Video, ISO/IEC 13818-2 and ITU-T Recommendation H.262, 1996
- [4] Video Coding for Low Bit Rate Communication, ITU-T Recommendation H.263, Feb. 1998.
- [5] Information Technology - Coding of Audio-Visual Objects - Part 2: Visual, ISO/IEC 14496-2, 1999.
- [6] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G.J. Sullivan, "Performance comparison of video coding standards using Lagrangian coder control," in *Proceedings of IEEE International Conference on Image Processing 2002*, vol. 2, pp501-504.
- [7] Y.-L. Lee and H. W. Park, "Loop filtering and post-filtering for low-bitrates moving picture coding," *Signal Processing: Image Commun.*, vol. 16, pp. 871–890, 2001.
- [8] S. D. Kim, J. Yi, H. M. Kim, and J. B. Ra, "A deblocking filter with two separate modes in block-based video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 156–160, Feb. 1999.

- [9] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 614- 619, Jul. 2003.
- [10] H.264/AVC reference software JM7.2, Jul. 2003
- [11] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C. Wang, T.-H. Chang, L.-G. Chen, "Architecture design for deblocking filter in H.264/JVT/AVC," *Proc. of Multimedia and Expo*, vol. 1, pp. 693 –696, Jul. 2003.
- [12] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ ISO/ IEC 14496-10 AVC), Mar. 2003.
- [13] H.264/AVC reference software JM8.2, Jul. 2004
- [14] Meng, B.; Au, O.C, "Fast intra-prediction mode selection for 4x4 blocks in H.264" in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal*, 2003., vol. 3, 6-10 pp.III - 389-92 ,April2003
- [15] Meng, B., Au, O.C., Chi-Wah Wong, Hong-Kwai Lam, "Efficient intra-prediction mode selection for 4x4 blocks in H.264" in *Proc. of Int. Conf. on Multimedia and Expo*, 2003, vol. 3 , 6-9 Pages:III - 521-4, July 2003
- [16] Feng PAN, Xiao LIN, Rahardja SUSANTO, Keng Pang LIM, Zheng Guo LI, Ge Nan FENG, Da Jun WU, and Si WU, "Fast Mode Decision for Intra Prediction," JVT-G013, 7th Meeting, Pattaya II, Thailand, 7-14 March, 2003.
- [17] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG "Performance comparison: H.26L intra coding vs. JPEG2000" Klagenfurt, Austria, 22-26 July, 2002, JVT-D039
- [18] T.-C. Wang, Y.-W. Huang, H.-C. Fang, and L.-G. Chen, "Parallel 4\_4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2003, pp. 800–803.

## 作者簡歷

姓名：鄭朝鐘

籍貫：台灣省台南市

學歷：

國立台南市第一高級中學 (民國 85 年 9 月～民國 88 年 6 月)

國立交通大學電子工程學系 學士 (民國 88 年 9 月～民國 92 年 6 月)

國立交通大學電子研究所系統組 碩士 (民國 92 年 9 月～民國 94 年 6 月)

獲獎紀錄：

- 九十三學年度 大學院校積體電路設計競賽 (IC Contest)  
研究所/大學部 標準單元式設計組(Cell-based) 優等
- Asia and South Pacific Design Automation Conference (ASP-DAC) 2005  
Best Award of Student Design Contest
- 九十二學年度 大學院校矽智產設計競賽(IP Contest)  
Star Video Motion Estimation Engine QME  
Soft IP 不定題組 特優
- 九十一學年度 殷之同電子實驗計畫獎學金  
專題名稱：Automatic generation of Area-Effective Bit-Serial FIR Filters
- 九十一學年度上學期(大四) 電子工程系書卷獎
- 九十學年度下學期(大三) 電子工程系書卷獎
- 九十學年度上學期(大三) 電子工程系書卷獎



著作：

Chao-Chung Cheng, Tian-Sheuan Chang, "Fast Three Step Intra Prediction Algorithm for 4x4 blocks in H.264," International Conference on Circuit and System (ISCAS) 2005

Chao-Chung Cheng, Tian-Sheuan Chang, "An Hardware Efficient Deblocking Filter for H.264/AVC," International Conference on Consumer Electronics (ICCE) 2005

Hao-Yun Chin, Chao-Chung Cheng, Yu-Kun Lin, and Tian-Sheuan Chang, "A Bandwidth Efficient Subsampling-based Block Matching Architecture for Motion Estimation," Asia and South Pacific Design Automation Conference (ASP-DAC) 2005

Chao-Chung Cheng, Yu-Jen Wang, Tian-Sheuan Chang, "A Fast Fractional Pel Motion Estimation Alogrithm for H.264/AVC," The 16th VLSI Design/CAD symposium 2005

