

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

適用於高解析度靜態影像與視訊應用之

H.264/MPEG-4 AVC 框內編解碼器設計

Design of H.264/MPEG-4 AVC Intra Codec for High  
Definition Size Still Image and Video Applications

研究生：古君偉

指導教授：張添烜 博士

中華民國九十五年七月

適用於高解析度靜態影像與視訊應用之  
H.264/MPEG-4 AVC 框內編解碼器設計

Design of H.264/MPEG-4 AVC Intra Codec for High  
Definition Size Still Image and Video Applications

研究生：古君偉

Student: Chun-Wei Ku

指導教授：張添烜 博士

Advisor: Dr. Tian-Sheuan Chang



A Thesis  
Submitted to Institute of Electronics  
College of Electrical Engineering and Computer Science  
National Chiao Tung University  
in Partial Fulfillment of the Requirements  
for Degree of Master of Science  
in  
Electronic Engineering  
July 2006  
Hsinchu, Taiwan, Republic of China

## 誌謝

首先，要感謝我的指導教授—張添烜博士，在研究所生涯中，除了給我許多支持與鼓勵之外，在研究上也常和我互相討論想法，解決我的困難與疑問。張教授的支援讓我無後顧之憂，得以專心致力於研究，才有這本論文的誕生。對於張教授的恩情，感激不盡。

同時也要謝謝我的口試委員們，交大電子李鎮宜主任，清大電機陳永昌教授，感謝各位在百忙之中抽空前來指導我，各位教授的寶貴意見讓本篇論文得以更加完備。

接著，我要感謝實驗室的夥伴。謝謝鄭朝鐘學長，帶領我進入視訊處理的領域，教導許多研究的技巧和寶貴的經驗，並加強我的設計功力，奠定往後比賽得獎或硬體設計的基礎。謝謝張彥中學長和林佑昆學長，給予課程或研究上的指導，讓研究能順利進行。謝謝王裕仁同學，和我一起參加兩屆 IC 競賽皆拿下不錯的成績，並常常交流彼此的想法。也謝謝蔡旻奇同學和余國亘同學，一起共同完成編碼器的設計。此外，要謝謝海珊學長、史彥芪學長、吳錦木同學、子筠、嘉俊、得瑋、私璟、英澤等學弟，你們的幫忙讓我的實驗室生活能順利渡過。所有的一切，都是我在交大的寶貴回憶。

最後，我要感謝我的家人們，我的父親、母親、弟弟，你們的默默支持，是我能夠完成學業的最大動力。

在此，我謹把這篇論文獻給所有愛我與我愛的人。

# 適用於高解析度靜態影像與視訊應用之 H.264/MPEG-4 AVC 框內編解碼器設計

研究生：古君偉

指導教授：張添烜博士

國立交通大學

電子研究所

## 摘要

近幾十年來，數位視訊科技已被廣泛地使用並成為生活中不可或缺的一部分。隨著數位訊號處理的發展，以及對較佳編碼效能的要求，H.264/AVC 被認為是次世代的國際視訊編碼標準。和早期的標準相比，在強大的編碼技術下，新的視訊標準可以明顯地降低資料量但仍維持視訊品質。在這些技術中，空間性的框內編碼是具有高編碼效率的新工具。高品質的編碼效率使得框內編碼不但適用於單張畫面的視訊編碼，也適用於靜態影像壓縮，甚至可以和最新的影像編碼標準 JPEG2000 相比擬。然而，因為複雜的編碼技術，框內編碼的運算複雜度也比之前的標準高的多。因此，如何減少複雜度並設計一個高效能的框內編碼器或解碼器，而不會造成太多的效能衰減，是個重要的課題。在本篇論文中，我們提供一個框內編解碼器和一個快速框內編碼器的兩個硬體實現來解決此問題。

首先，我們提出一個演算法層次和系統層次皆最佳化的基本規格框內編解碼器架構。為了在近似相同的視訊品質下減少硬體成本和增加處理速度，以硬體為目的的演算法移除了佔空間的平面預測並以更準確的代價函數來加強模式決定過程。在架構設計方面，除了快速的模組實現外，由巨圖塊層次的管線化型式和三

個排程技術來安排編碼過程，以避免閒置的週期並改善資料生產量。整個編解碼器設計最後可以分別在 117MHz 時脈下支援高解析度 1280x720 尺寸 30fps 的即時視訊編碼，以及在 58MHz 下支援高解析度 1920x1080 尺寸的視訊解碼。

另一個成果，是具有快速模式決定演算法和可變像素平行化技術，針對低功率問題設計的基本規格框內編碼器。經由提出修改後的三步驟演算法流程，模式決定的過程可以被縮短。此外，可變像素平行化的資料路徑也可以有效地節省約一半處理週期，並導致較低的頻率需求。在交錯排程的技術和三個低功率考量的策略下，新設計比之前的設計有較小的晶片面積，並只需 61MHz 即可支援高畫質 1280x720 尺寸 30fps 的即時視訊編碼。

簡而言之，我們對於 H.264/AVC 框內編碼的貢獻可以分成兩個部分。一個貢獻是框內編解碼器，在最小的硬體成本和處理速度的改進下，整合了編碼和解碼的過程。另一個貢獻是快速框內編碼器，特性包括了降低運算複雜度，壓制頻率需求，以及對於低功率課題的策略。



# **Design of H.264/MPEG-4 AVC Intra Codec for High Definition Size Still Image and Video Applications**

Student: Chen-Wei Ku

Advisor: Dr. Tian-Sheuan Chang

Institute of Electronics

National Chiao Tung University

## **Abstract**

For the recent decades, digital video technology has been popularly used and become a necessary part in our daily life. With the development of digital signal processing and demand of better coding performance, H.264/AVC is regarded as the international video coding standard for the next generation. The new standard can achieve significant bitrate reduction compared to earlier standards but still maintains the video quality with its powerful coding techniques. In these techniques, the spatial intra coding is a newly proposed coding tool with high coding efficiency. The high-quality coding efficiency makes intra coding not only suitable for single-picture video coding but also for still image compression, and even competitive with the latest image coding standard like JPEG2000. However, due to the complicated coding techniques, computational complexity of intra coding is much higher than previous standards as well. Thus, how to reduce the complexity and to design a high-efficient intra coder or decoder without much performance degradation is an important issue. In this thesis, we contribute two hardware implementations of an intra frame codec and a fast intra frame encoder to solve this question.

We first propose a baseline intra frame codec architecture with both algorithm-level

and system-level optimization. To reduce hardware cost and increase processing speed while providing nearly the same video quality, the hardware-oriented algorithm removes the area-costly plane prediction and enhances the mode decision process with more accurate cost function. In the architecture design, in addition to fast module implementation the process is arranged by the macroblock-level pipelining style together with three scheduling techniques to avoid idle cycles and improve data throughput. The whole codec design finally can support high definition 1280x720 size 30fps real-time video coding at 30fps when clocked at 117MHz and high definition 1920x1080 size decoding at 58MHz respectively.

The other work is the baseline intra frame encoder targeted on low-power issues with techniques like fast mode decision algorithm and variable-pixel parallelism. The mode decision process is shortened by the proposed modified three-step algorithm. Besides, the variable-pixel parallel datapath can also effectively save almost half of processing cycles and lead to lower frequency requirement. With the technique of interlaced scheduling and three strategies for low-power consideration, the new design has smaller chip area relative to previous designs and can support high definition 1280x720 size 30fps real-time video coding at only 61MHz.

In brief, our contributions to H.264/AVC intra coding can be divided into two parts. One contribution is the intra frame codec, which integrates both encoding and decoding processes with minor hardware cost and improvement of processing speed. The other contribution is the fast intra frame encoder, with features of reduction of computational complexity, suppression of frequency requirement, and strategies for low-power issues.

# Content

<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1. Motivation .....	1
1.2. Thesis Organization .....	4
<b>Chapter 2 Overview of H.264/AVC Standard</b> .....	<b>5</b>
2.1. Fundamental of H.264/AVC .....	5
2.1.1. Coding Structure .....	5
2.1.2. Features of Standard .....	7
2.1.3. Profiles.....	10
2.2. Components of Baseline Intra Coding .....	11
2.2.1. Intra Prediction .....	11
2.2.2. Cost Generation and Mode Decision.....	12
2.2.3. Transform .....	14
2.2.4. Quantization .....	15
2.2.5. Entropy Coding .....	15
<b>Chapter 3 H.264/AVC Intra Frame Codec</b> .....	<b>18</b>
3.1. Hardware Oriented Algorithm.....	18
3.1.1. Enhanced SATD Function for Mode Decision.....	18
3.1.2. Intra Plane Mode Removal.....	20
3.1.3. Simulation Results.....	22
3.2. System Level Scheme.....	27
3.2.1. Analysis of Hardware Complexity .....	27



3.2.2. Macroblock Level Pipelining .....	29
3.3. Architecture Design of Intra Codec .....	30
3.3.1. Overall Architecture .....	30
3.3.2. Schedule of Codec .....	33
3.3.3. Intra Prediction Generation Unit .....	35
3.3.4. Transform Unit .....	37
3.3.5. Quantization and De-quantization .....	38
3.3.6. Cost Generation and Mode Decision Unit.....	39
3.3.7. Reconstruction Path.....	40
3.3.8. Memory Organization.....	41
3.3.9. CAVLC Codec .....	42
3.4. Implementation Results .....	44
3.4.1. Gate-count and Layout .....	44
3.4.2. Comparison.....	46
<b>Chapter 4 Fast H.264/AVC Intra Frame Encoder .....</b>	<b>48</b>
4.1. Fast Algorithm for Intra Prediction .....	48
4.1.1. Survey of Fast Algorithm .....	48
4.1.2. Modified Fast Algorithm for Intra Prediction .....	50
4.1.3. Simulation Results.....	53
4.2. Architecture Design of Fast Intra Encoder .....	57
4.2.1. Overall Architecture .....	57
4.2.2. Scheduling of Encoder .....	59
4.2.3. Eight-pixel Parallel Datapath.....	60
4.2.4. Memory Organization.....	62

4.2.5. Strategies for Low Power Design.....	63
4.3. Implementation Results .....	65
4.3.1. Gate-count and Layout .....	65
4.3.2. Comparison.....	67
<b>Chapter 5 Conclusion.....</b>	<b>69</b>



## List of Figures

Fig. 1	Hierarchy of video data components .....	6
Fig. 2	Basic structure diagram of H.264/AVC encoder .....	7
Fig. 3	Basic structure diagram of H.264/AVC decoder .....	7
Fig. 4	Three profiles of H.264/AVC .....	10
Fig. 5	Nine modes for intra 4x4 prediction.....	12
Fig. 6	Four modes for Intra 16x16 or 8x8 prediction .....	12
Fig. 7	Flow diagram of most probable mode selection.....	13
Fig. 8	(a) Prefix and suffix bitstrings, (b) exp-Golomb bitstrings, (c) mapping for signed bitstrings.....	16
Fig. 9	Example of CAVLC Coding.....	17
Fig. 10	Four categorized types of intra prediction modes .....	21
Fig. 11	Intra plane mode for (a) 16x16 (b) 8x8 predictions.....	22
Fig. 12	RD curves of [10] and proposed algorithm for sequence “Stefan” .....	24
Fig. 13	RD curves of [10] and proposed algorithm for sequence “Mobile” .....	24
Fig. 14	RD curves of [10] and proposed algorithm for sequence “Paris” .....	25
Fig. 15	RD curves of [10] and proposed algorithm for sequence “Akiyo” .....	25
Fig. 16	RD curves of [10] and proposed algorithm for sequence “Foreman” .....	26
Fig. 17	RD curves of [10] and proposed algorithm for sequence “Coastguard” .....	26
Fig. 18	Ping-pong architecture with macroblock level pipelining for encoder .....	29
Fig. 19	Proposed architecture of baseline intra frame codec .....	30
Fig. 20	Encoder dataflow of the design .....	31
Fig. 21	Decoder dataflow of the design.....	32

Fig. 22	Pipelined schedule for codec design.....	35
Fig. 23	Reconfigurable datapath of intra prediction generation unit.....	35
Fig. 24	Examples of operations for four intra prediction modes .....	36
Fig. 25	Hardware architecture of transform unit [20].....	38
Fig. 26	Quantization and de-quantization unit.....	39
Fig. 27	Cost generation and mode decision unit.....	40
Fig. 28	Memory organization in intra codec.....	41
Fig. 29	Architecture for CAVLC encoder.....	42
Fig. 30	Architecture for CAVLC decoder.....	43
Fig. 31	Layout of the codec chip .....	45
Fig. 32	Decision flow of fast three-step algorithm for intra prediction.....	50
Fig. 33	Fast three-step algorithm in pipeline structure .....	51
Fig. 34	Decision flow of modified three-step algorithm for intra prediction .....	52
Fig. 35	Modified fast three-step algorithm in pipeline structure .....	52
Fig. 36	RD curves of [10] and proposed fast algorithm for sequence “Stefan” .....	54
Fig. 37	RD curves of [10] and proposed fast algorithm for sequence “Mobile” .....	55
Fig. 38	RD curves of [10] and proposed fast algorithm for sequence “Paris” .....	55
Fig. 39	RD curves of [10] and proposed fast algorithm for sequence “Akiyo” .....	56
Fig. 40	RD curves of [10] and proposed fast algorithm for sequence “Foreman” .....	56
Fig. 41	RD curves of [10] and proposed fast algorithm for sequence “Coastguard” .....	57
Fig. 42	Proposed architecture of encoder with fast algorithm.....	58
Fig. 43	Pipelined schedule for fast encoder when best luma mode is selected to 16x16 .....	60
Fig. 44	Pipelined schedule for fast encoder when best luma mode is selected to 4x4 60	

Fig. 45	Eight-pixel parallel intra prediction generator.....	61
Fig. 46	Eight-input eight-output 4x4 transform unit.....	62
Fig. 47	Memory organization in fast encoder.....	63
Fig. 48	Layout of the fast encoder chip .....	66



## List of Tables

Table 1	Average bitrate saving for video streaming.....	3
Table 2	Quantization factors in H.264/AVC.....	15
Table 3	De-quantization factors in H.264/AVC .....	15
Table 4	Probability distribution of 16x16 modes in different sequence with 300 I-frames at QP=28 .....	22
Table 5	Comparison among original code [10], SAITD algorithm in [14], and the two proposed algorithm for coding of 300 Intra frames.....	23
Table 6	Data throughput for different video size.....	27
Table 7	Frequency for N-pixel parallel encoder.....	28
Table 8	Frequency for N-pixel parallel decoder.....	28
Table 9	List of gate count for proposed design.....	44
Table 10	Information for the chip.....	45
Table 11	Comparison among [13], [21], and this work.....	46
Table 12	Comparison among original [10], modified algorithm, and proposed fast algorithm combined of three techniques for 300 Intra frames .....	53
Table 13	List of gate count for fast encoder.....	65
Table 14	Information for the encoder chip.....	66
Table 15	Comparison among previous codec in Chapter 3 , [13], and this work .....	67

# Chapter 1

## Introduction

With the demand of higher video quality and lower bitrate, and feasibility of fast growing semiconductor processing, a new video coding standard is developed by the Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG as the next generation video compression standard, which is known as H.264 or MPEG-4 Part 10 Advanced Video Coding (AVC) [1]. In comparison with existing video standards such as MPEG-2 [2] and MPEG-4 [3], the latest standard can improve the coding efficiency by up to 50% while still keep the video quality [4] with various newly introduced coding tools.

### 1.1. Motivation

Earlier video coding standard MPEG-2, also known as H.262, was developed in the last decade as the extension of prior MPEG-1 [5]. This standard is still popularly used in many applications of our daily life, such as transmission of TV signals, satellite communication, cable, and storage of high-quality video signal on DVDs. However, with the popularity of high definition TV and flexibility of multimedia services, present standard cannot afford the requirement of video quality and real-life applications. In addition, transmission capacity in some transmission media such as cable modem and DSL is much lower than in broadcast channels, which also limits the utilization of MPEG-2.

For videoconference and multimedia streaming service, H.263 [6] and its later enhancement H.263+ [7] were evolved by ITU-T to deal with the low-bitrate video coding in telecommunication application. The other standard MPEG-4 was also

launched in the recent years to address the future multimedia applications like interactive TV and internet video. The MPEG-4 standard consists of more parts besides traditional video and audio system. Its video standard allows coding, scalability, and access to an individual object, and can achieve better coding efficiency relative to prior standards. Since MPEG-4 video standard is built on the same coding structure of MPEG-2 and added with new coding tools, it offers modest coding gain at the expense of a modest increase in complexity [4]. As a result, the increase of complexity should be only justified for object-based video coding but not for nature rectangular video applications.

To formulate a new standard of next generation video coding, the joint team of ITU-T VCEG and ISO MPEG was established to co-develop it for natural video. The newest international video coding standard, well-known as H.264/AVC, is approved by ITU-T as Recommendation H.264 and by ISO/IEC as International Standard 14496-10 MPEG4 Part 10 Advanced Video Coding. The H.264/AVC is designed for technical solution of various application areas, for example, broadcast system over cable or satellite, internet video, interactive storage on optical devices, wireless and mobile network, and multimedia streaming service. To satisfy the flexibility of multiple applications, H.264/AVC can adjust the coding complexity depending on the different profiles defined in the standard.

The coding architecture of H.264/AVC standard is different with that of existing MPEG-2 standard and has noticeable improvement in both bitrate decrease and preservation of decoded video quality [8]. Table 1 presents the average bitrate saving of four popular video standards, where the bitrate decrease in H.264/AVC relative to MPEG-4, H.263, and MPEG-2 are 39%, 49%, and 64% respectively.



Table 1 Average bitrate saving for video streaming

Coder	Average bit-rate savings relative to:		
	MPEG-4 ASP	H.263 HLP	MPEG-2
JVT	39%	49%	64%
MPEG-4 ASP	-	17%	43%
H.263 HLP	-	-	31%

The significant improvement in H.264/AVC is caused by various enhanced and new coding techniques, which can achieve higher video quality and better compression rate than any other standard. These techniques include variable block size and multiple reference pictures motion estimation/compensation, simplified small block size integer transform, quarter-sample accuracy in motion vectors, in-loop deblocking filter, directional spatial-domain intra prediction, context adaptive entropy coding, and arithmetic entropy coding.

In the above mentioned techniques, spatial-domain intra prediction is a new feature in H.264/AVC. Previous standard, MPEG-4, uses the Intra-DC and Intra-AC technique to encode an I-frame with just encoding the difference between neighboring transformed blocks. The newly introduced intra coding method in H.264/AVC takes advantage of the relationship of correlation among adjacent blocks to reduce data correlation of the encoding picture. It predicts the currently coded block with pixel values from neighboring blocks with various directions and only encodes the residues. The use of spatial-domain prediction is able to achieve higher coding efficiency in the single frame coding than that in the previous standards, and even competitive with the latest still image coding standard, JPEG2000 [9].

As a result, the intra frame only coding and decoding is very suitable for applications that do not need or cannot afford the inter prediction capability. The hardware of intra frame codec can be used in portable consumer products like digital video recorder or

digital still camera. For this application, a design with low power consumption and low memory cost are also necessary to fit the demand of practicability.

## **1.2. Thesis Organization**

This paper is organized with five parts. Chapter 1 gives the introduction and motivation of this work. Then in Chapter 2 , a brief overview of H.264/AVC standard and its intra coding is given. Chapter 3 presents an architecture design of H.264/AVC intra frame codec and the hardware oriented algorithm. In Chapter 4 a proposed intra frame encoder with fast prediction technique and lower working frequency is implemented. Finally a conclusion remark is given in Chapter 5 .



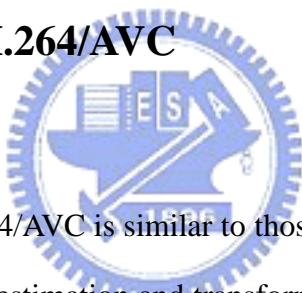
## Chapter 2

# Overview of H.264/AVC Standard

Earlier standards like MPEG-1 and MPEG-2 have enabled many popular consumer products such as video CDs and DVDs. As their successor, H.264/AVC is created more powerful in coding performance and more flexible in all kinds of applications. With the highly developed signal processing and semiconductor technology, many complicated and computationally intensive coding tools can be supported efficiently in H.264/AVC standard. These progressive tools have ability to improve the coding efficiency obviously but still maintain the decoded video quality.

### 2.1. Fundamental of H.264/AVC

#### 2.1.1. Coding Structure



The coding structure of H.264/AVC is similar to those of previous standards and built on the commonly used motion estimation and transform coding structure. This standard supports the 4:2:0 format with 8-bit sample precision for pixel values and encodes the video by picture order. A picture, as well as an interlaced field or a non-interlaced frame, can be partitioned into several slices, and each slice consists of a series of macroblocks. A macroblock is a primary unit for video coding and includes one 16x16 luminance (luma) and two 8x8 chrominance (chroma) components. The macroblock can further be separated into 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 sub-macroblocks for motion estimation/compensation, where the 4x4 one is called a block. The hierarchy of data organization in H.264/AVC is shown in Fig. 1.

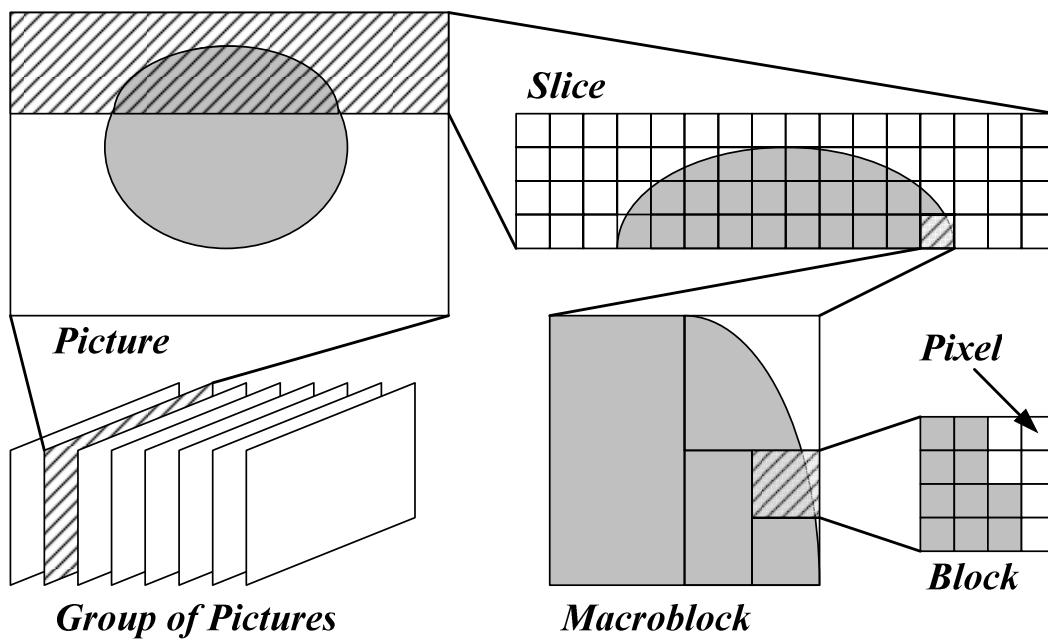


Fig. 1 Hierarchy of video data components

Fig. 2 shows the basic structure diagram of H.264/AVC encoder, and Fig. 3 shows the decoder. The encoding flow first performs the intra prediction in I-slice or motion estimation in P-slice from some reference pictures. Residual values, the difference between predicted values and original ones, are then sent to forward transform unit and quantized after inter or intra prediction. The quantized coefficients, syntax, motion vectors, and other coding information are further coded by entropy coding. In addition, the quantized coefficients are also reconstructed through de-quantization, scaling, inverse transform, and motion compensation as the reference for next slice processing. To decode a slice, residual values are recovered through entropy decoding, de-quantization, and inverse transform in proper order. Decoded slices can be reconstructed by adding residual values to established data from motion compensation for P-slice or intra mode prediction for I-slice. Detailed coding flow can refer to [1] or [4].

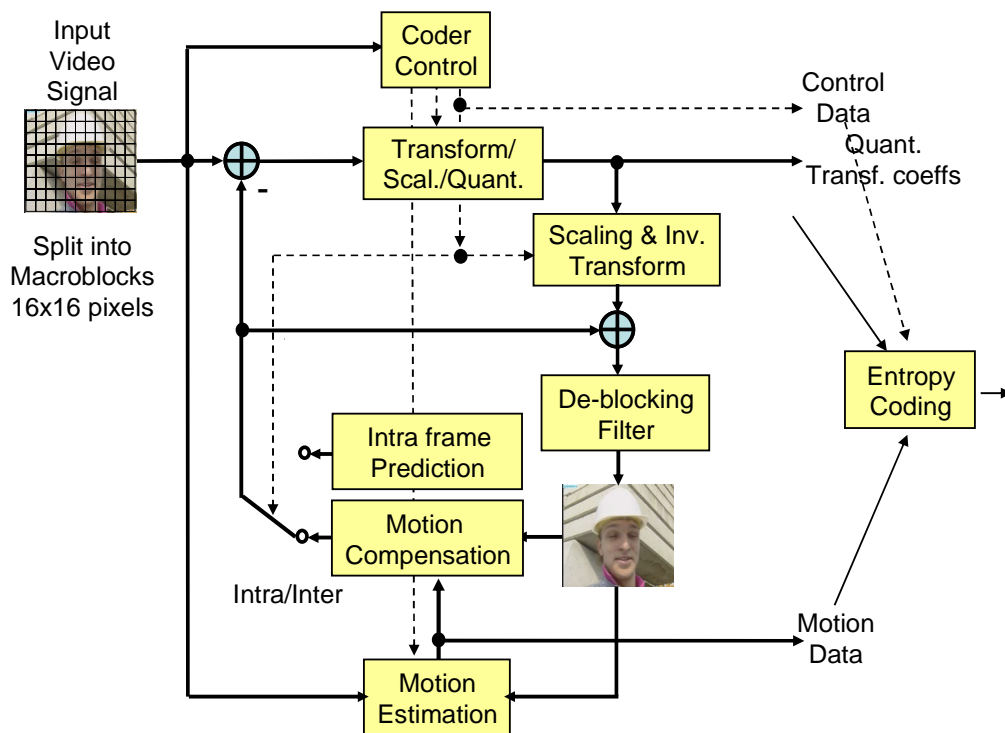


Fig. 2 Basic structure diagram of H.264/AVC encoder

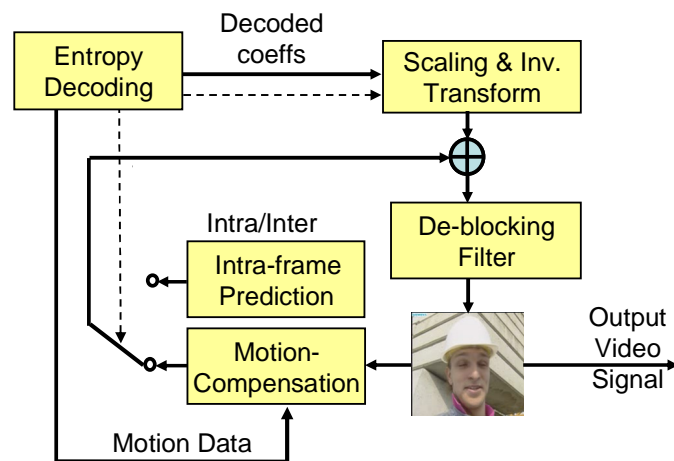


Fig. 3 Basic structure diagram of H.264/AVC decoder

### 2.1.2. Features of Standard

The H.264/AVC standard adopts many new coding tools which are never introduced in the earlier standards, and it also enhances previous techniques to obtain better coding

efficiency. These features significantly improve the coding performance and make the standard satisfy every technical application such as broadcast system and internet video. Some of the important features in H.264/AVC are introduced in the following.

### **1. Variable block-size motion estimation and compensation**

The standard has more flexibility in selection of block sizes and shapes for motion estimation and compensation than any previous standard. Seven kinds of block sizes are introduced, including 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4. This helps to enhance the efficiency of coding of irregularly shaped objects or background behind moving objects.

### **2. Quarter-sample-accurate motion vector**

Most of the previous standards enable half-sample motion vector accuracy, but H.264/AVC improves it by adding quarter-sample motion vector accuracy, which is first found in the advanced profile of the MPEG-4 Visual (Part 2) standard. However, this standard further reduces the complexity of the interpolation processing to simplify the computation.

### **3. Multiple reference picture motion estimation and compensation**

In MPEG-2 and its following standards, only one previous picture can be used to predict the values in the incoming picture. The H.264/AVC standard enlarges the selection range of reference pictures from one to more for better coding efficiency. Thus, motion vectors across multiple reference pictures are allowed. In addition, the standard also allows the bi-direction prediction coding which uses both previous and next pictures as reference ones.

#### **4. Spatial-based directional intra prediction coding**

In previous standards, such as MPEG-1 and MPEG-2, I-pictures are directly coded. MPEG-4 Visual standard [3] adopts the Intra-AC and Intra-DC prediction for coding of I-pictures, which utilizes neighboring transformed blocks to perform the prediction and residual coding. However, these coding methods do not take advantage of the correlation among adjacent neighboring blocks. Thus, a spatial-based prediction technique with directional pixel mapping is presented in H.264/AVC for I-picture coding before transform, which uses the reconstructed neighboring pixels to perform the prediction with modes from different directions. With this technique, the coding efficiency for I-pictures can be improved effectively.

#### **5. Small block size integer transform**

All of the prior video standards use a transform block size of 8x8, while the new H.264/AVC uses a smaller transform size of 4x4. This allows the encoder to represent signals in a more locally-adaptive fashion and reduces the artifacts caused by the edges of different pixels.

#### **6. In-loop deblocking filter**

Block-based video coding may raise the blocking artifacts due to both prediction and residual difference coding of the decoding process. The solution to this problem is to use an adaptive deblocking filter which can improve the resulting video quality well. Instead of building as an optical feature in H.263+, in H.264/AVC the deblocking filter is positioned in the motion compensation loop as an in-loop filter so that quality improvement in a single picture can be extended to the inter-picture prediction as well.

#### **7. Context-adaptive entropy coding**

For compression of quantized transform coefficients, an efficient variable-length coding (VLC) method is used in H.264/AVC. The VLC coding is previously used in the existing standards but enhanced in this standard with context adaptivity to increase the coding performance.

## 8. Arithmetic entropy coding

Another coding method known as context-adaptive binary arithmetic coding (CABAC) is also included in H.264/AVC as the advanced entropy coding. This arithmetic coding can achieve higher efficiency than VLC coding due to the effective probability model of symbol occurrence.

### 2.1.3. Profiles

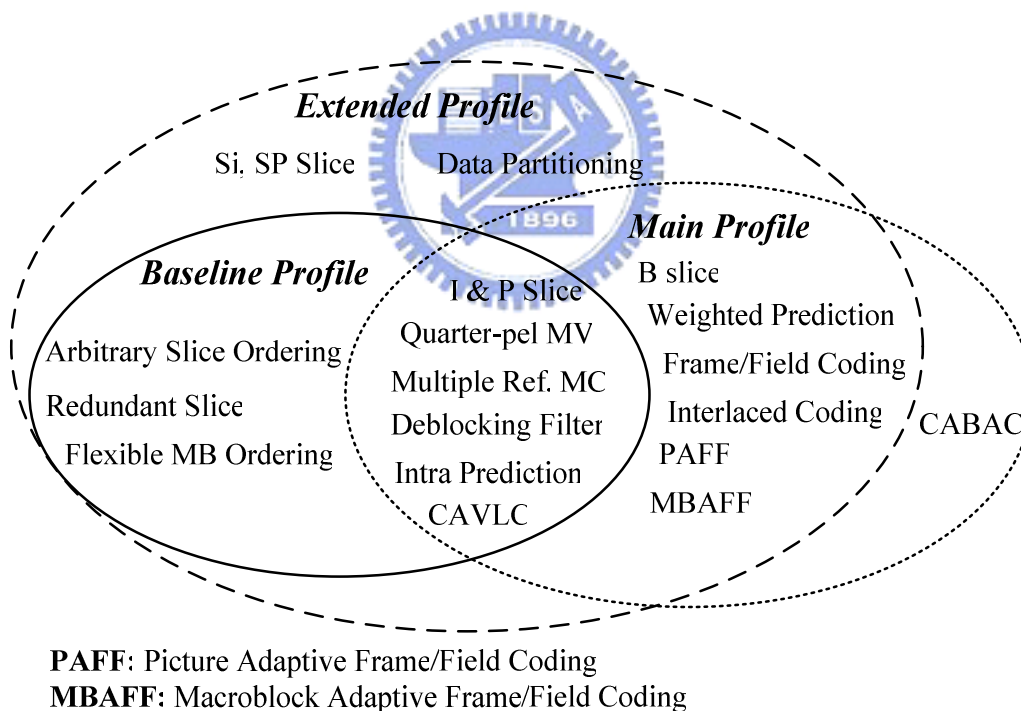


Fig. 4 Three profiles of H.264/AVC

There are three profiles defined in H.264/AVC as shown in Fig. 4, which are baseline, main, and extended profiles. Baseline profile includes basic coding tools and features,



such as I-slice, P-slice, quarter-sample accurate motion vector, deblocking filter, and CAVLC, and is primarily used for lower-cost applications with demand of less computing resources. Thus, this profile is widely used in videoconferencing, internet multimedia, and mobile applications. Main profile is used as the mainstream consumer profile for applications of broadcast system and storage devices. It contains most of the features in baseline profile and other advanced techniques, like adaptive frame/field coding, interlaced coding, weighted prediction, B-slice, and CABAC. The main profile can achieve better performance in both bitrate saving and video quality while needs much more computation effort. Finally the extended profile, which includes all the features in baseline profile and main profile except CABAC, is intended as the streaming video profile and has relatively high compression capability with extra tricks for robustness to data losses and server stream switching.

## **2.2. Components of Baseline Intra Coding**

### **2.2.1. Intra Prediction**

Spatial-domain prediction is the main feature of H.264/AVC intra coding. There are two kinds of intra prediction for luma components, nine 4x4 prediction modes or four 16x16 prediction modes, which are shown in Fig. 5 and Fig. 6 respectively. The 4x4 prediction modes use the neighboring thirteen reconstructed samples denoted from A to M in Fig. 5 to predict the block pixels with eight different directions and one average value. For 16x16 prediction modes, the values are predicted from the 32 adjacent boundary pixels of upper and left macroblocks. Similar procedures are also applied to the chroma components where four 8x8 prediction modes are used with 16 neighboring pixels.

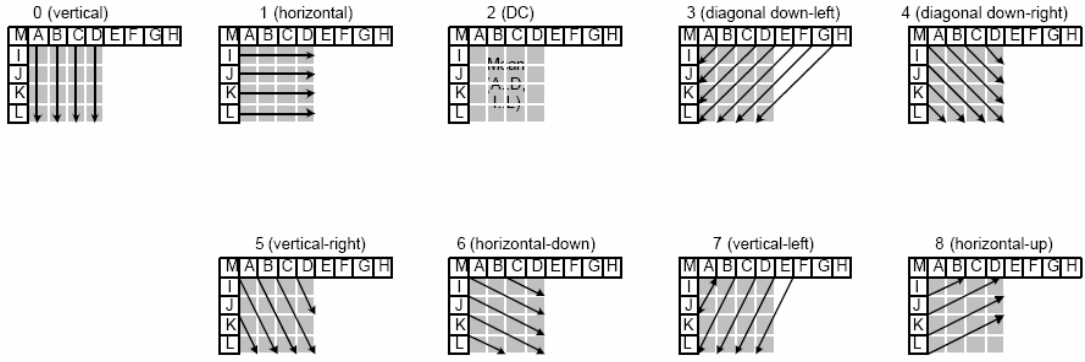


Fig. 5 Nine modes for intra 4x4 prediction

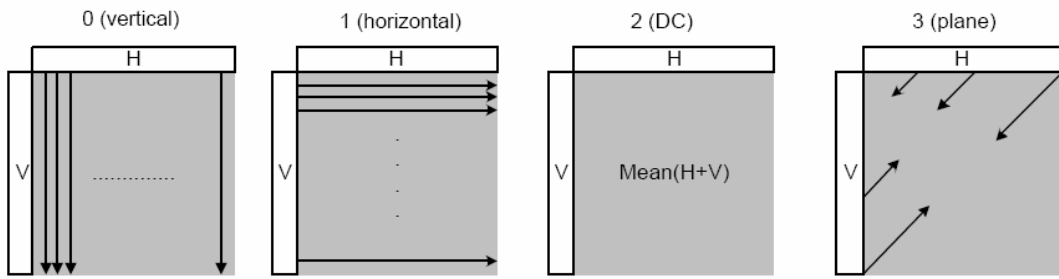


Fig. 6 Four modes for Intra 16x16 or 8x8 prediction

### 2.2.2. Cost Generation and Mode Decision

The best mode decision for intra prediction in [10] can be either the time consuming rate distortion optimization (RDO) or just much simpler cost accumulation. RDO uses the weighted sum of actual encoded bitrate and the reconstructed samples to produce distortion. Though it can achieve better performance, it is computationally intensive.

An alternative way is using cost accumulation. Two generally used mode decision methods for cost generation are available in [10], sum of absolute difference (SAD) and sum of absolute transform difference (SATD). The formulas are defined as follows.

$$C_1 = SAD + 4m\lambda(Q_p) \quad (1)$$

$$C_2 = SATD + 4m\lambda(Q_p) \quad (2)$$

$$SAD = \sum_{i=1}^4 \sum_{j=1}^4 |s_{ij} - p_{ij}| \quad (3)$$

$$SATD = \sum_{i=1}^4 \sum_{j=1}^4 |T(s_{ij} - p_{ij})| \quad (4)$$

In (1) and (2), symbol  $\lambda(Qp)$  stands for the lambda values which are from the approximated exponential function depending on the quantization parameters, and the variable  $m$  indicate whether the current mode is the most probable mode. If the most probable mode is detected,  $m$  is equal to 0, otherwise it sets to 1. The decision flow of most probable is illustrated in Fig. 7. In (3) and (4),  $s_{ij}$  and  $p_{ij}$  are the  $(i, j)$ th elements of source block and predicted block respectively. The function  $T(x)$  in (4) represents the 4x4 discrete Hadamard transform (DHT), as shown in (5), where symbol  $X$  indicates the residual block.

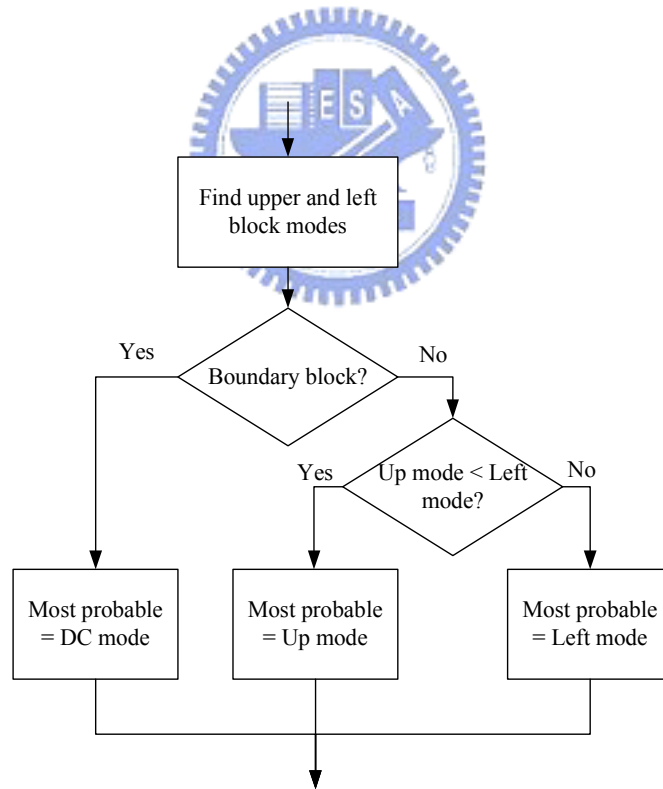


Fig. 7 Flow diagram of most probable mode selection

$$T(X) = \left( \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} X \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2 \quad (5)$$

For every four blocks in Z-scan order of a macroblock, another lambda value,  $6\lambda(Qp)+0.5$ , is added to cost value for adjusting the weight. The best mode is finally decided by comparing the summarized cost value of sixteen blocks in the 4x4 prediction to the best mode of the 16x16 prediction.

### 2.2.3. Transform

In H.264/AVC, a traditional 8x8 floating-point discrete cosine transform (DCT) in previous standards is replaced by an approximated 4x4 DCT. The transform can be divided into two parts, 4x4 integer transform and fractional scalar multiplication factors that are further merged into the quantization stage. The forward 4x4 integer transform with quantization is illustrated in (6), and the inverse one with de-quantization is illustrated in (7). In which, the matrices with factors  $a$  and  $b$  denote the scalar multiplication. Notice that matrix  $X$  in (6) is post-scaled by quantization and matrix  $Y$  in (7) is pre-scaled by de-quantization. For a macroblock predicted by the 16x16 or 8x8 modes, the DC value of each transformed block is further processed by 4x4 DHT or 2x2 DHT respectively, as shown in (5) or (8)

$$Y = \left( \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} X \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} \right) \quad (6)$$

$$X_R = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left( \begin{bmatrix} Y \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \quad (7)$$

$$T_c(X) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} X \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8)$$

### 2.2.4. Quantization

In the quantization stage, there are 52 values of quantization parameters (QPs) and corresponding quantization steps supplied in H.264/AVC standard. The steps are doubled for increase of every six numbers in QPs. The scaling factors mentioned in Subsection 2.2.3 are incorporated into these quantization factors to avoid the computational complexity in 4x4 transform stage. These factors for quantization and de-quantization are implemented in [10] as a multiplication factors and shown in Table 2 and Table 3 respectively.

Table 2 Quantization factors in 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

Table 3 De-quantization factors in 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	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

### 2.2.5. Entropy Coding

There are two entropy coding methods supported in H.264/AVC baseline profile.

#### 1. Exp-Golomb coding



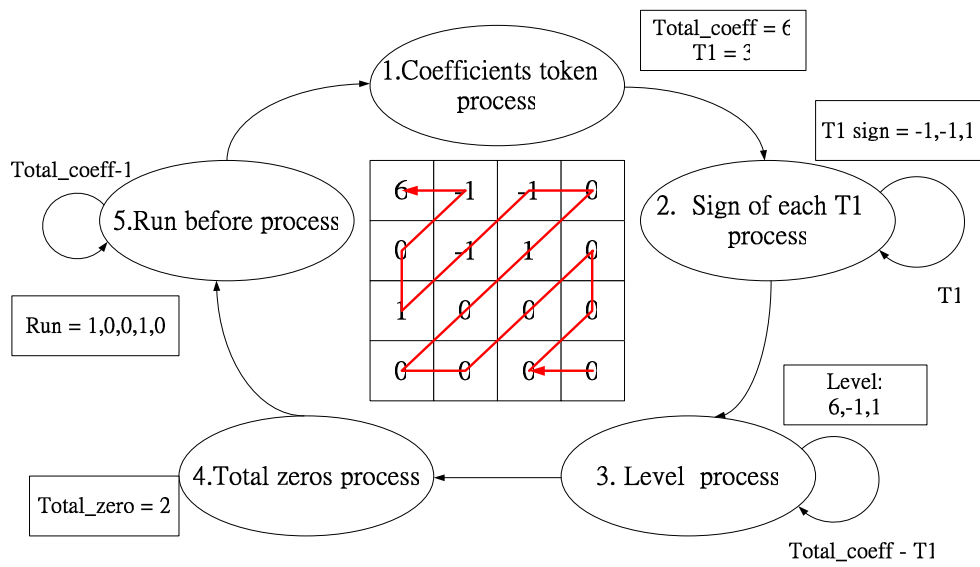


Fig. 9 Example of CAVLC Coding



## Chapter 3

### H.264/AVC Intra Frame Codec

H.264/AVC is popularly regarded as the video standard in the next generation to replace the existing MPEG-2 standards. The spatial-domain intra coding is a newly supported technique which is not only suitable for moving video coding but also still image compression. However, the common digital signal processors are hard to afford its high computational complexity and large data throughput.

In this chapter, a parallel H.264/MPEG-4 AVC baseline profile intra frame codec supporting both processes of encoder and decoder is proposed for digital camera and video application. This work is mainly based on the previous architecture [11] and modified to fit the decoding procedure. The proposed chip has ability to support high definition (HD) size 720p (1280x720 4:2:0) at 30 fps real-time video encoding at 117MHz and 1080p (1920x1080 4:2:0) at 30 fps video decoding at 58MHz. When clocked at the height frequency 125MHz, this design can process encoding of 29.62M pixels still image per second or decoding of 135.60M pixels. The research result of this work is also published in [12].

#### 3.1. Hardware Oriented Algorithm

##### 3.1.1. Enhanced SATD Function for Mode Decision

In determining the coding performance of intra-only H.264/AVC, the cost function for mode decision is the most important part. To find a best matched prediction mode is to use RDO. Though RDO can provide the best performance, its complexity hinders its use in the hardware design. Thus, the SATD method is adopted to calculate the costs.



However, how to determine the transform for SATD computation will become the main issue now. The transform choice used in SATD should be computationally simple but also effective to estimate the energy of the signals. In [10], a pure transform of 4x4 DHT is adopted for mode decision, but it is far from the real transform used in the whole encoding process. A better transform choice for SATD shall approximate the effect of transform and quantization used in the H.264/AVC encoding process to estimate the real bitrate. Therefore, previous works [13][14] use the 4x4 integer transform as the choice. Although their approaches can achieve better performance than DHT does, it's still not good enough. That is because that the fractional multiplication factors do not be taken into consideration. A complete transform function in H.264/AVC shall include both the integer transform and multiplication factors in the quantization formula as shown in (6) and (7). However, to incorporate these factors into the cost function directly will cost a lot of computation because they are not simple integer numbers. Besides, these factors cannot be directly derived from the formula since the quantization parameters shall also be included.

To solve these problems, this work adopts the cost function proposed in [11] that combines the integer transform and simplified multiplication factors. The simplified multiplication factors are derived from quantization coefficients shown in Table 2 and Table 3 . Derivation from the quantization coefficients enables the consideration of both effects of transform and quantization. From these tables, we can obtain the required scalar factors by approximating the relationship among the reciprocal of de-quantization coefficients and simplifying them to integers for reduction of computational complexity as shown in (9) and (10).

$$1/\text{quant\_coef}: \quad p(0,0)^{-1} : p(0,1)^{-1} : p(1,1)^{-1} \approx 30:19:12 \quad (9)$$

$$1/\text{dequant\_coef}: \quad p(0,0)^{-1} : p(0,1)^{-1} : p(1,1)^{-1} \sim= 30:25:20 \quad (10)$$

In (9) and (10) the symbol  $p(x,y)$  represents the quantization and de-quantization coefficients of different positions in Table 2 and Table 3 respectively. The scaling factors derived from the de-quantization table are adopted by considering the final performance and implementation cost, as shown in (11). In this formula, division by 32 is added to avoid enlargement of cost values, which can be carried out with simpler low-cost wiring in the hardware design. As a result, the cost generation function is able to estimate the energy of residuals after the transform and quantization function more accurate than other methods while still keep computation simple and suitability for hardware implementation. It can provide better quality than that in [10] and can be used to compensate the quality loss of the plane mode removal discussed in Subsection 3.1.2.

$$C(X) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \begin{bmatrix} X \\ \phantom{X} \\ \phantom{X} \\ \phantom{X} \end{bmatrix} \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{pmatrix} \otimes \begin{bmatrix} 32 & 25 & 32 & 25 \\ 25 & 20 & 25 & 20 \\ 32 & 25 & 32 & 25 \\ 25 & 20 & 25 & 20 \end{bmatrix} / 32 \quad (11)$$

### 3.1.2. Intra Plane Mode Removal

The various intra prediction modes can further be organized systematically into four types according to their prediction properties and computational complexity. These types are illustrated in Fig. 10. The bypass type is easy to be implemented since the prediction samples are the same as boundary pixels. In the average type, neighboring eight pixels (for 4x4 prediction) or 32 pixels (for 16x16 prediction) are summarized and divided into an average value for all prediction samples. The linear type contains most of the 4x4 prediction modes with directional approach, and the samples are linearly interpolated by boundary pixels. Finally in the bilinear type, also known as plane prediction, samples are derived by the approximation of bilinear transform. Though

being simplified to be only integer arithmetic operations, the plane mode is still much more computational complex than other modes. Besides, it is also hard to reuse its results for other prediction, and occupies almost half of the area in the intra prediction unit. The detail computation of plane prediction can refer to Fig. 11.

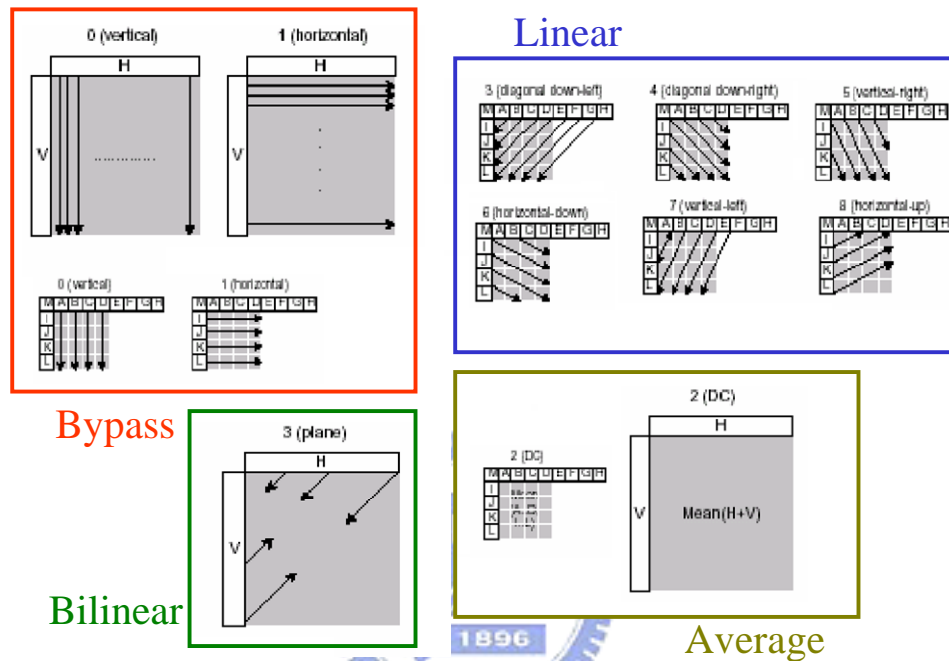
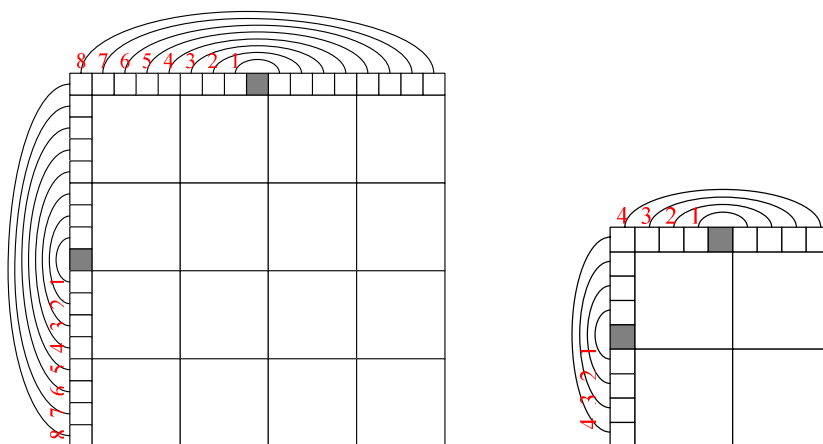


Fig. 10 Four categorized types of intra prediction modes

A solution to this problem is to eliminate the plane mode from intra prediction and replace it with other modes. This may raise the issue of performance loss. Table 4 shows the probability distribution of the 16x16 prediction modes in different sequences. Macroblocks predicted in plane mode is only 4.2% in average and not larger than 5.7% except the sequence “Akiyo” which contains much smoother texture. However, after simulation we found that prediction with plane mode only reduces about 1% of bitrate than that without plane mode for these video sequences. This 1% bitrate loss can be easily compensated by the enhanced cost function proposed in Subsection 3.1.1. With the modification, we can achieve almost the same result as [10] but save a lot of

computation and hardware cost.



#### Luma 16x16

$$h = \sum x [ p(7+i,-1) - p(7-i,-1) ] \quad i=1\sim 8$$

$$v = \sum y [ p(-1,7+i) - p(-1,7-i) ] \quad i=1\sim 8$$

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

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

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

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

(a)

#### Chroma 8x8

$$h = \sum x [ p(4+i,-1) - p(4-i,-1) ] \quad i=1\sim 4$$

$$v = \sum y [ p(-1,4+i) - p(-1,4-i) ] \quad i=1\sim 4$$

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

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

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

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

(b)

Fig. 11 Intra plane mode for (a) 16x16 (b) 8x8 predictions

Table 4 Probability distribution of 16x16 modes in different sequence with 300 I-frames at QP=28

Sequence	16x16 modes				
	Total ratio	Vertical	Horizontal	DC	Plane
Mobile	3.3%	0.8%	1.0%	1.3%	0.2%
Coastguard	10.7%	0.8%	3.8%	4.6%	1.6%
Stefan	20.7%	3.4%	12.8%	2.0%	2.5%
Paris	15.5%	3.2%	4.6%	4.4%	3.4%
Foreman	23.1%	5.1%	4.3%	8.0%	5.7%
Akiyo	47.5%	5.3%	4.8%	25.7%	11.8%

### 3.1.3. Simulation Results

Table 5 illustrates the comparison results for encoding of six CIF-size sequences with all intra frames in different QPs among four algorithms: the original SATD function in [10], SAITD algorithm in [14], enhanced SATD cost function, and the proposed method

combining enhanced function and plane mode removal. In most cases of the simulation, it is obvious that the proposed SATD cost function is able to achieve better coding efficiency than [10] and [14], with almost the same or even better PSNR quality. We can also observe that the enhanced algorithm can reduce average 0.08% bitrate for all sequences. After combining with technique of plane mode removal, the bitrate increase is compensated and not larger than 0.06% in average.

Table 5 Comparison among original code [10], SAITD algorithm in [14], and the two proposed algorithm for coding of 300 Intra frames

Sequence	QP	JM 8.6 [9]				SAITD Algorithm				Enhanced SATD Cost Function				Enhanced SATD Cost Function + Plane Mode Removal			
		SNR Y	SNR U	SNR V	Bit-rate	SNR Y	SNR U	SNR V	Bit-rate	SNR Y	SNR U	SNR V	Bit-rate	SNR Y	SNR U	SNR V	Bit-rate
Stefan	16	46.38	47.27	47.43	10537.41	+0.10	+0.00	+0.00	+0.11%	+0.06	+0.00	-0.01	-0.16%	+0.06	+0.00	+0.00	-0.11%
	20	42.96	44.43	44.51	8143.18	+0.05	-0.03	-0.03	+0.07%	+0.02	-0.03	-0.03	-0.18%	+0.02	-0.03	-0.03	-0.12%
	24	39.63	41.63	41.65	6189.86	-0.02	-0.14	-0.13	+0.15%	-0.04	-0.14	-0.13	-0.22%	-0.04	-0.14	-0.12	-0.14%
	28	36.41	38.95	38.96	4585.29	-0.02	-0.23	-0.24	+0.15%	-0.05	-0.23	-0.24	-0.24%	-0.05	-0.23	-0.25	-0.16%
	32	33.05	37.12	37.08	3246.41	-0.05	-0.41	-0.43	+0.34%	-0.08	-0.41	-0.43	-0.21%	-0.08	-0.42	-0.45	-0.15%
36	29.96	35.15	35.07	2218.46	-0.06	-0.48	-0.52	+0.95%	-0.10	-0.49	-0.52	-0.09%	-0.11	-0.53	-0.55	-0.06%	
Mobile	16	45.93	46.25	46.29	15361.27	+0.04	+0.01	+0.01	+0.05%	+0.04	+0.01	+0.01	-0.10%	+0.04	+0.01	+0.01	-0.08%
	20	42.14	42.87	42.89	12233.46	+0.05	-0.03	-0.04	+0.07%	+0.03	-0.03	-0.04	-0.12%	+0.03	-0.03	-0.03	-0.09%
	24	38.49	39.72	39.68	9487.00	+0.01	-0.11	-0.10	+0.17%	-0.01	-0.11	-0.10	-0.14%	-0.01	-0.11	-0.10	-0.10%
	28	35.04	36.88	36.76	7179.38	-0.03	-0.17	-0.18	+0.26%	+0.01	-0.17	-0.18	-0.16%	+0.01	-0.16	-0.18	-0.12%
	32	31.50	34.89	34.67	5200.07	+0.02	-0.24	-0.24	+0.52%	-0.01	-0.24	-0.24	-0.14%	-0.01	-0.24	-0.23	-0.09%
36	28.28	32.87	32.60	3572.40	+0.01	-0.31	-0.32	+1.08%	-0.05	-0.31	-0.22	-0.05%	-0.05	-0.31	-0.22	+0.01%	
Paris	16	46.16	47.35	47.63	10114.93	+0.08	-0.05	-0.05	+0.12%	+0.06	-0.05	-0.05	-0.09%	+0.06	-0.05	-0.04	-0.00%
	20	42.81	44.69	44.88	7662.99	+0.02	-0.21	-0.16	+0.25%	-0.01	-0.21	-0.16	-0.08%	-0.01	-0.20	-0.15	-0.01%
	24	39.59	41.97	42.12	5742.80	-0.05	-0.35	-0.28	+0.49%	-0.09	-0.36	-0.28	-0.06%	-0.09	-0.36	-0.28	+0.05%
	28	36.49	39.40	39.54	4235.43	-0.05	-0.50	-0.34	+0.70%	-0.20	-0.50	-0.34	-0.06%	-0.10	-0.52	-0.36	+0.09%
	32	33.33	37.51	37.73	3005.78	-0.14	-0.61	-0.52	+0.82%	-0.20	-0.61	-0.52	-0.11%	-0.20	-0.65	-0.52	+0.08%
36	30.36	35.58	35.78	2055.30	-0.18	-0.64	-0.54	+1.00%	-0.25	-0.64	-0.55	-0.13%	-0.25	-0.67	-0.56	+0.08%	
Akiyo	16	47.34	48.46	49.40	4159.54	+0.16	-0.06	-0.11	+0.95%	+0.09	-0.05	-0.12	-0.10%	+0.10	-0.06	-0.12	+0.42%
	20	44.89	46.88	47.95	2777.52	+0.04	-0.26	-0.33	+0.83%	-0.05	-0.26	-0.34	-0.16%	-0.05	-0.30	-0.38	+0.18%
	24	42.65	44.62	46.03	1941.72	-0.14	-0.35	-0.58	+1.03%	-0.23	-0.35	-0.58	+0.18%	-0.25	-0.37	-0.64	+0.56%
	28	40.33	42.52	43.93	1370.30	-0.18	-0.53	-0.77	+1.49%	-0.33	-0.54	-0.76	+0.27%	-0.35	-0.63	-0.85	+0.62%
	32	37.77	40.82	42.54	963.42	-0.32	-0.53	-0.89	+1.71%	-0.53	-0.52	-0.86	-0.66%	-0.58	-0.69	-1.09	+0.30%
36	35.28	38.80	40.69	672.96	-0.38	-0.45	-0.78	+2.80%	-0.65	-0.50	-0.75	+0.17%	-0.70	-0.64	-0.98	+0.28%	
Foreman	16	46.26	47.56	48.57	7665.68	+0.10	+0.03	-0.01	+0.28%	+0.08	+0.03	-0.01	-0.14%	+0.08	+0.03	+0.00	-0.04%
	20	42.90	45.17	46.83	5394.72	+0.13	-0.08	-0.21	+0.59%	+0.09	-0.08	-0.21	-0.13%	+0.09	-0.08	-0.21	-0.00%
	24	39.95	42.87	44.78	3678.28	+0.04	-0.25	-0.33	+1.20%	-0.03	-0.24	-0.33	-0.08%	-0.03	-0.25	-0.35	+0.07%
	28	37.26	40.91	42.79	2467.35	+0.03	-0.32	-0.42	+1.80%	-0.06	-0.32	-0.42	-0.08%	-0.06	-0.34	-0.46	+0.10%
	32	34.61	39.80	41.32	1598.24	-0.05	-0.41	-0.54	+2.62%	-0.19	-0.41	-0.54	+0.03%	-0.20	-0.46	-0.60	+0.30%
36	32.24	38.61	39.81	1022.62	-0.08	-0.39	-0.49	+4.19%	-0.31	-0.39	-0.49	-0.00%	-0.32	-0.45	-0.57	+0.53%	
Coastguard	16	45.88	48.20	49.05	9454.85	+0.04	+0.06	+0.05	+0.15%	+0.04	+0.06	+0.05	-0.20%	+0.04	+0.06	+0.05	-0.18%
	20	42.13	46.38	47.64	6969.04	+0.10	-0.04	-0.10	+0.30%	+0.09	-0.04	-0.10	-0.21%	+0.09	-0.05	-0.11	-0.20%
	24	38.74	44.64	46.13	4959.03	+0.08	-0.21	-0.12	+0.74%	+0.06	-0.21	-0.11	-0.16%	+0.06	-0.24	-0.13	-0.15%
	28	35.63	43.08	44.72	3437.66	+0.15	-0.27	-0.15	+1.33%	+0.12	-0.27	-0.15	-0.11%	+0.12	-0.35	-0.18	-0.14%
	32	32.74	41.96	43.72	2236.41	+0.06	-0.31	-0.17	+1.99%	+0.00	-0.31	-0.16	-0.13%	+0.00	-0.44	-0.23	-0.17%
36	30.24	40.82	42.72	1418.33	-0.04	-0.28	-0.11	+2.68%	-0.15	-0.27	-0.11	-0.17%	-0.14	-0.39	-0.14	-0.03%	

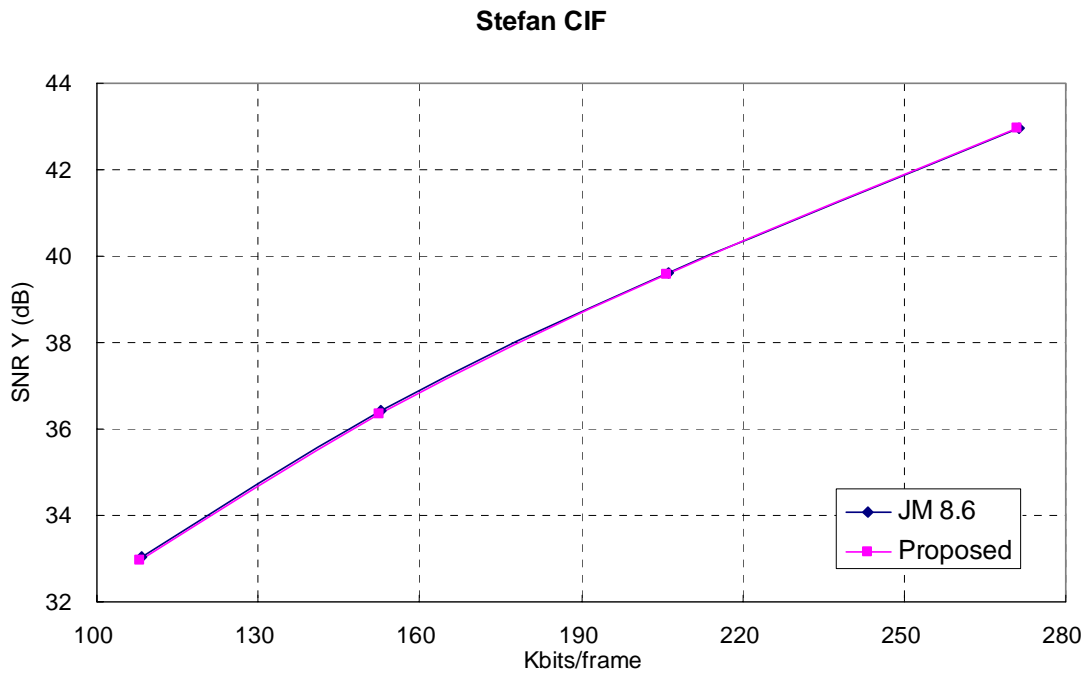


Fig. 12 RD curves of [10] and proposed algorithm for sequence “Stefan”

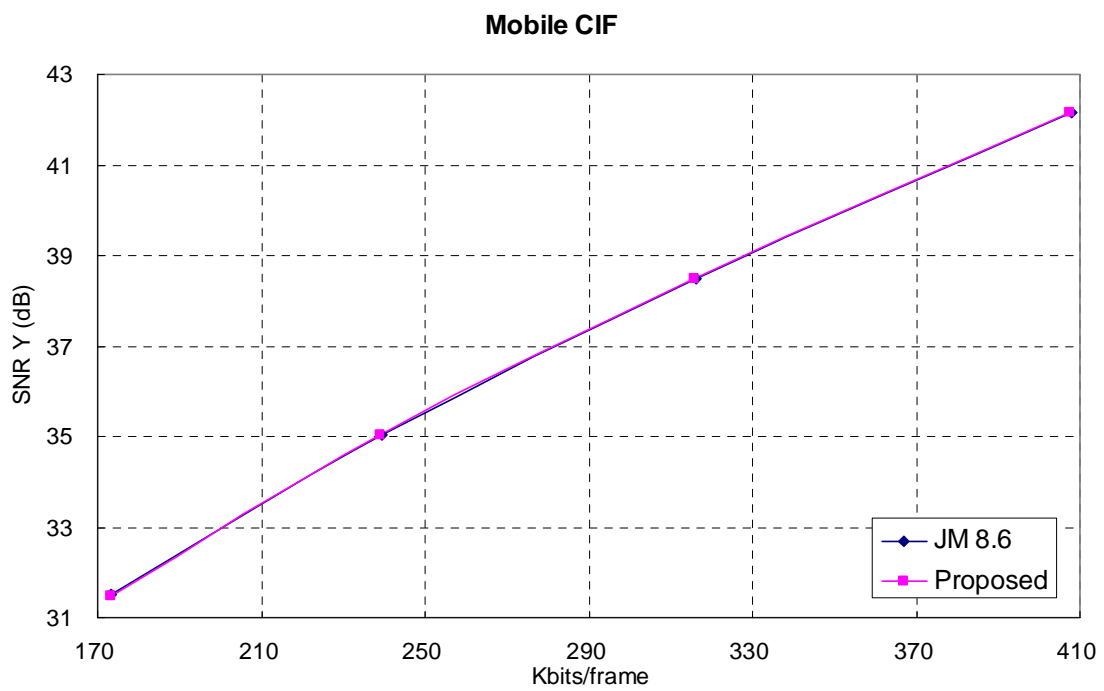


Fig. 13 RD curves of [10] and proposed algorithm for sequence “Mobile”

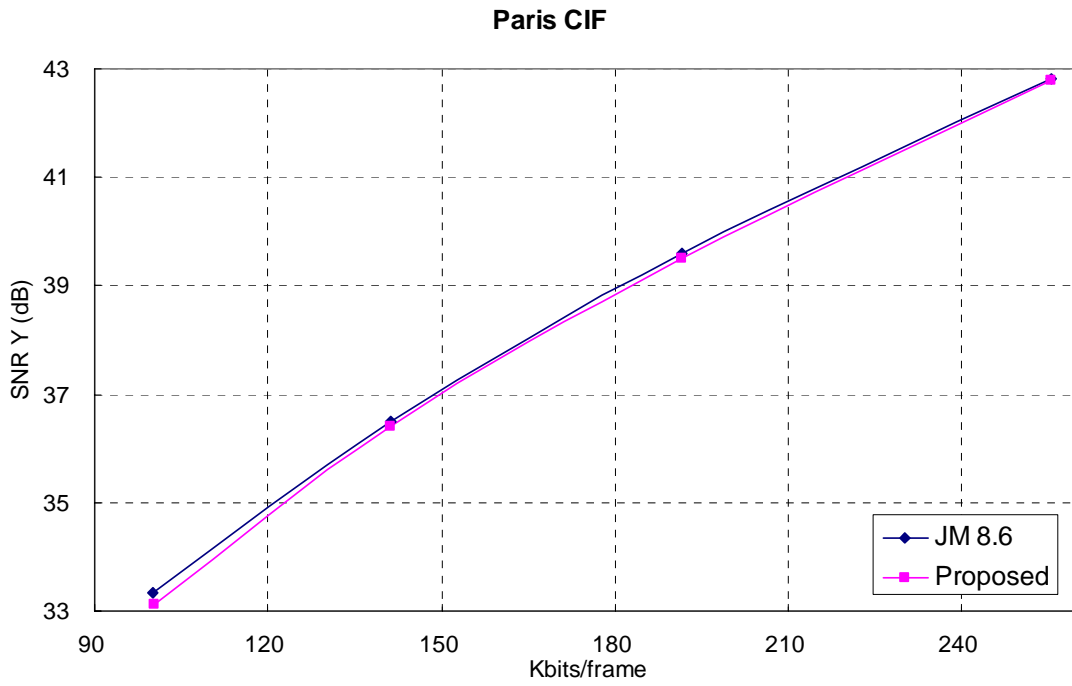


Fig. 14 RD curves of [10] and proposed algorithm for sequence “Paris”

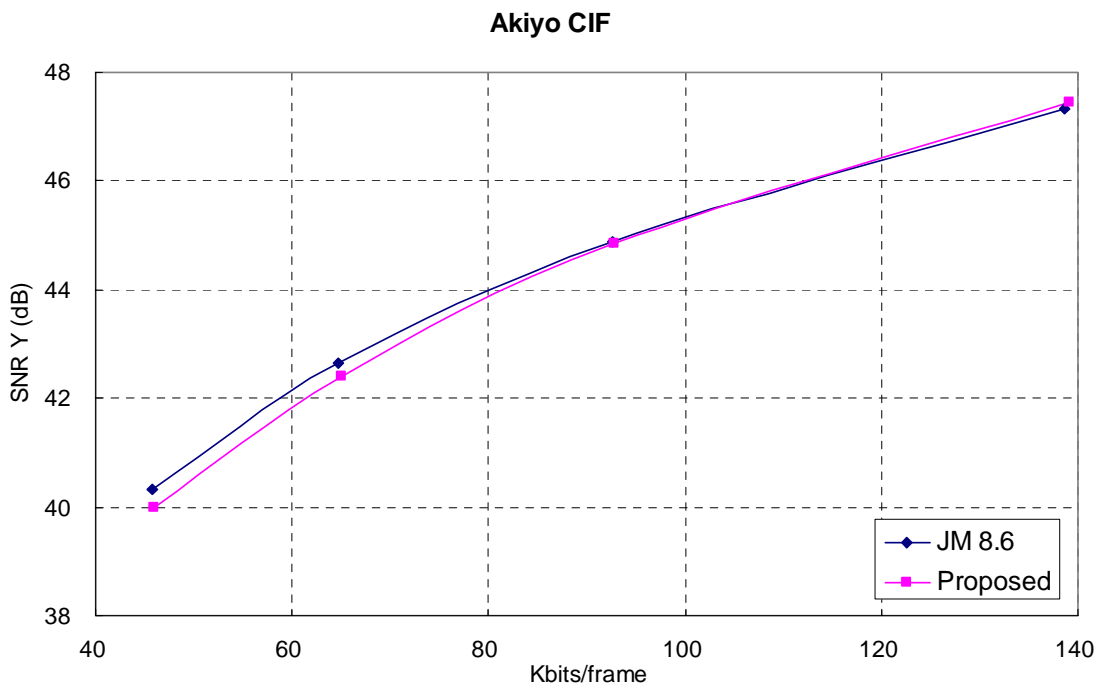


Fig. 15 RD curves of [10] and proposed algorithm for sequence “Akiyo”

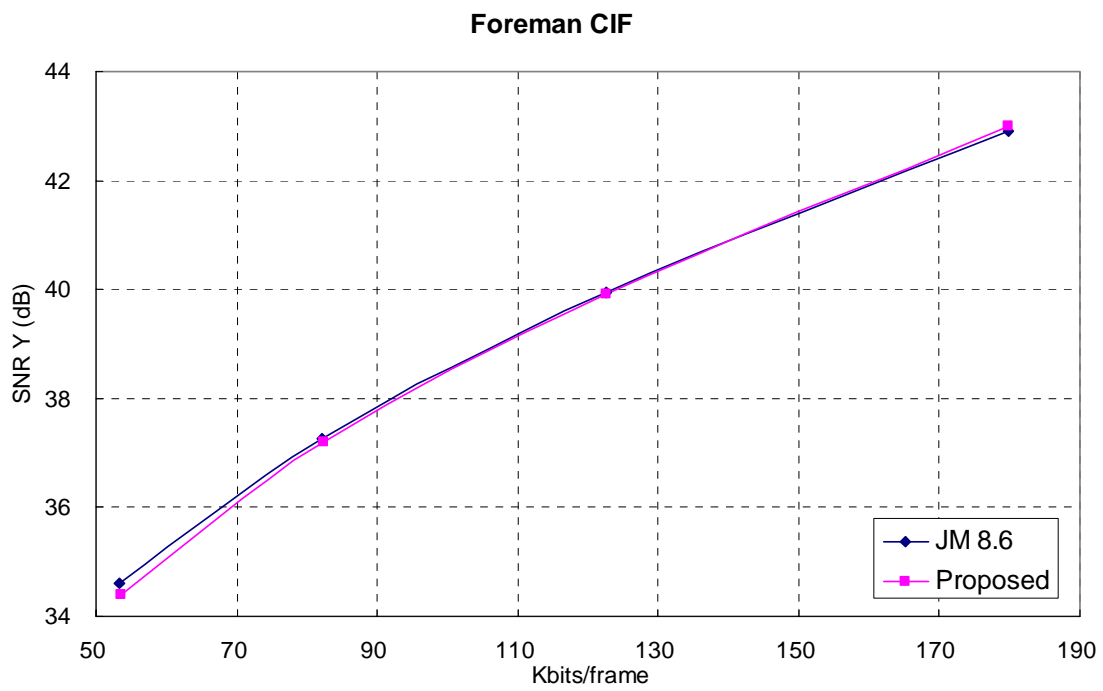


Fig. 16 RD curves of [10] and proposed algorithm for sequence “Foreman”

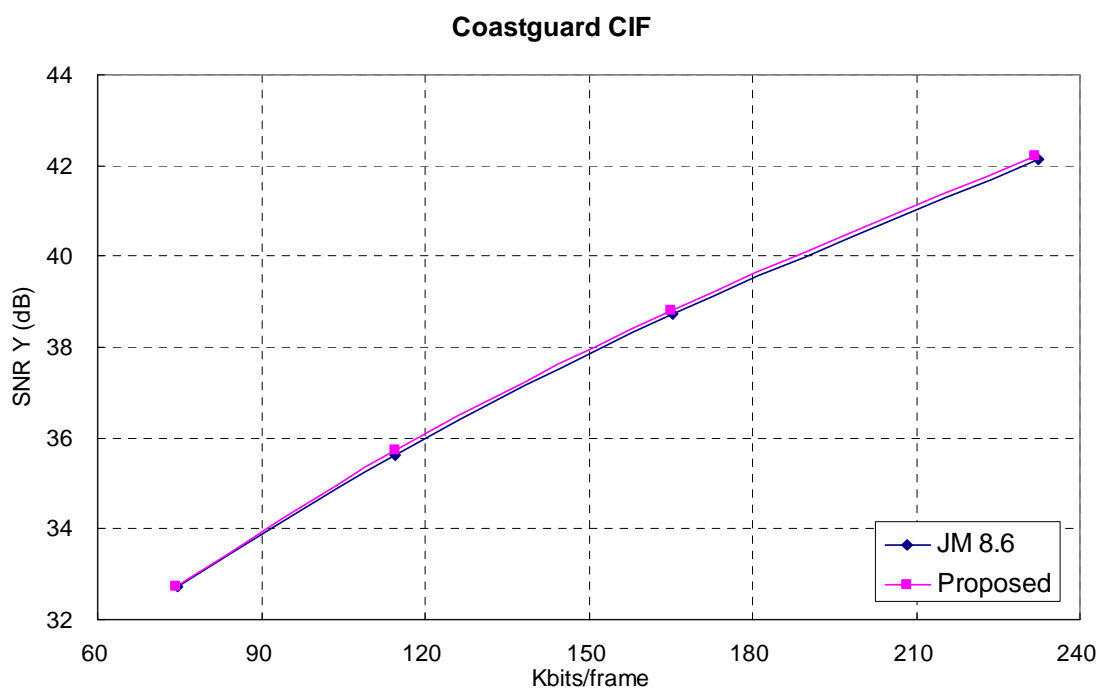


Fig. 17 RD curves of [10] and proposed algorithm for sequence “Coastguard”



The RD curve diagrams of [10] and our proposed combined algorithm for these six sequences are shown from Fig. 12 to Fig. 17. The QP range for these diagrams is from 20 to 32 except that for “Akiyo” whose range is located from 16 to 28 to clearly show the characteristic of its curve. The curves of our algorithm are very close to the original ones. Especially in the high bitrate coding with lower QPs, the performance is even better. This algorithm-level optimization actually makes the final hardware design not only simpler but also with good video quality.

## 3.2. System Level Scheme

### 3.2.1. Analysis of Hardware Complexity

To achieve the throughput for our target of video size, the complexity of hardware shall be first analyzed before design. For H.264/AVC codec, the computational complexity in encoder is much more extensive than that in decoder since encoder computes all prediction modes instead of decoding exactly one. Table 6 shows the data throughput in different video sizes at 30 fps. Thus, for our target HD 720p, it needs data throughput of at least 108,000 macroblocks per second, which is identical to 27.65M pixels.

Table 6 Data throughput for different video size

Video Size		Data Throughput	
		Mega pixs/sec	kilo mbs/sec
QCIF	176 x 144	0.76	2.97
CIF	352 x 288	3.04	11.88
ITU-R	720 x 576	12.44	48.60
SDTV	720 x 480	10.37	40.50
HDTV	1280 x 720	27.65	108.00
	1920 x 1080	62.21	243.00

To estimate the operating frequency for hardware design, we explore the cycles for

intra coding. Simplifying the estimation by neglecting the cycles of data transfer between on-chip and off-chip memory, we only consider prediction cycles. With such assumption, total cycle count in encoding process to predict a macroblock, including one luma and two chroma components, is 3456 ( $16 \times 16 \times 9 + 16 \times 16 \times 3 + 2 \times 16 \times 4 \times 3$ ), where plane modes are removed and other operations are excluded. The necessary frequency is 373.25 MHz for HD 720p size and 839.81 MHz for HD 1080p size in response to the estimated cycles in encoder. This speed requirement is far beyond the generally acceptable range of common processor and hard to be implemented.

Table 7 Frequency for N-pixel parallel encoder

Video Size		Frequency at N-Parallel (MHz)			
		N=1	N=2	N=4	N=16
QCIF	176 x 144	10.26	5.13	2.57	0.64
CIF	352 x 288	41.06	20.53	10.26	2.57
ITU-R	720 x 576	167.96	83.98	41.99	10.50
SDTV	720 x 480	139.97	69.98	34.99	8.75
HDTV	1280 x 720	373.25	186.62	<b>93.31</b>	23.33
	1920 x 1080	839.81	419.90	209.95	52.49

Table 8 Frequency for N-pixel parallel decoder

Video Size		Frequency at N-Parallel (MHz)			
		N=1	N=2	N=4	N=16
QCIF	176 x 144	1.14	0.57	0.29	0.07
CIF	352 x 288	4.56	2.28	1.14	0.29
ITU-R	720 x 576	18.66	9.33	4.67	1.17
SDTV	720 x 480	15.55	7.78	3.89	0.97
HDTV	1280 x 720	41.47	20.74	<b>10.37</b>	2.59
	1920 x 1080	93.31	46.66	23.33	5.83

As a consequence, we apply parallelism technique to reduce the required frequency. Table 7 and Table 8 show the estimation results of encoder and decoder respectively for such pixel parallelism. For encoder design, the suitable choice is to use the four-pixel parallel architecture for HD 720p that runs at frequency of 93.31MHz and needs 864

cycles for one macroblock. With the same condition, the four-pixel parallel decoder only needs 96 cycles at 10.37MHz to decode a macroblock for 720p size. Thus, our decoder can support larger size like HD 1080p at 23.33MHz. Such design target can achieve the real-time requirement while is easy to be implemented as well.

### 3.2.2. Macroblock Level Pipelining

Previous approach in Subsection 3.2.1 only assumes the cycles for intra prediction. However, more cycles will be required when considering other functions like data transfer between memories and entropy coding. These operations will increase the necessity of extra cycle count for a macroblock and result in higher operating frequency. For example, the CAVLC circuit in [15] takes about 500 cycles to encode a high-quality application video. This will increase the encoder latency to around 1,400 cycles with the frequency of 150MHz. In addition, the zigzag scan in CAVLC unit will also increase the operation cycles since its scan order is quite different than the raster scan used in the prediction engine.

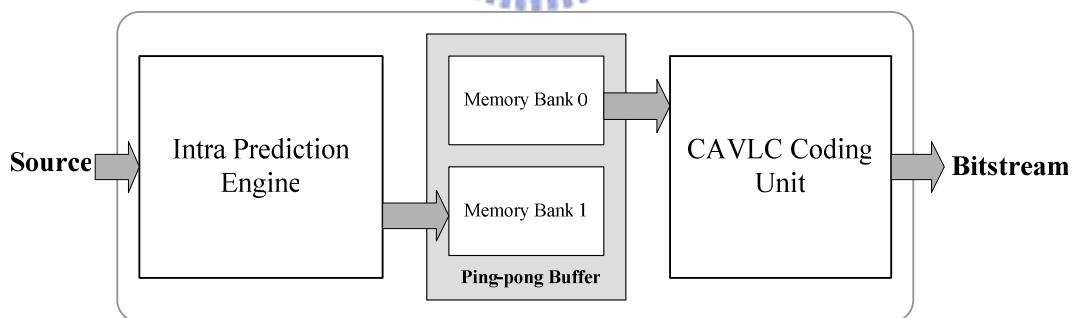


Fig. 18 Ping-pong architecture with macroblock level pipelining for encoder

To solve above problems, the macroblock level pipelining is used as shown in Fig. 18. This pipeline partition enables the overlapped execution of intra prediction and entropy coding without large cycle increases. The cycle count of a macroblock depends on the longest latency of each processing unit in pipeline. Besides, this design adopts the

ping-pong architecture with a two-bank memory located between the prediction loop and entropy coding to resolve the ordering problem. In the architecture, currently predicted coefficients after quantization are sent to one memory bank of ping-pong buffer, and coefficients of previously predicted macroblock are stored in the other memory bank and ready for CAVLC coding. These ping-pong buffers are also beneficial for decoding process in the inverse data flow direction for data reordering and processing rate smoothing.

### 3.3. Architecture Design of Intra Codec

#### 3.3.1. Overall Architecture

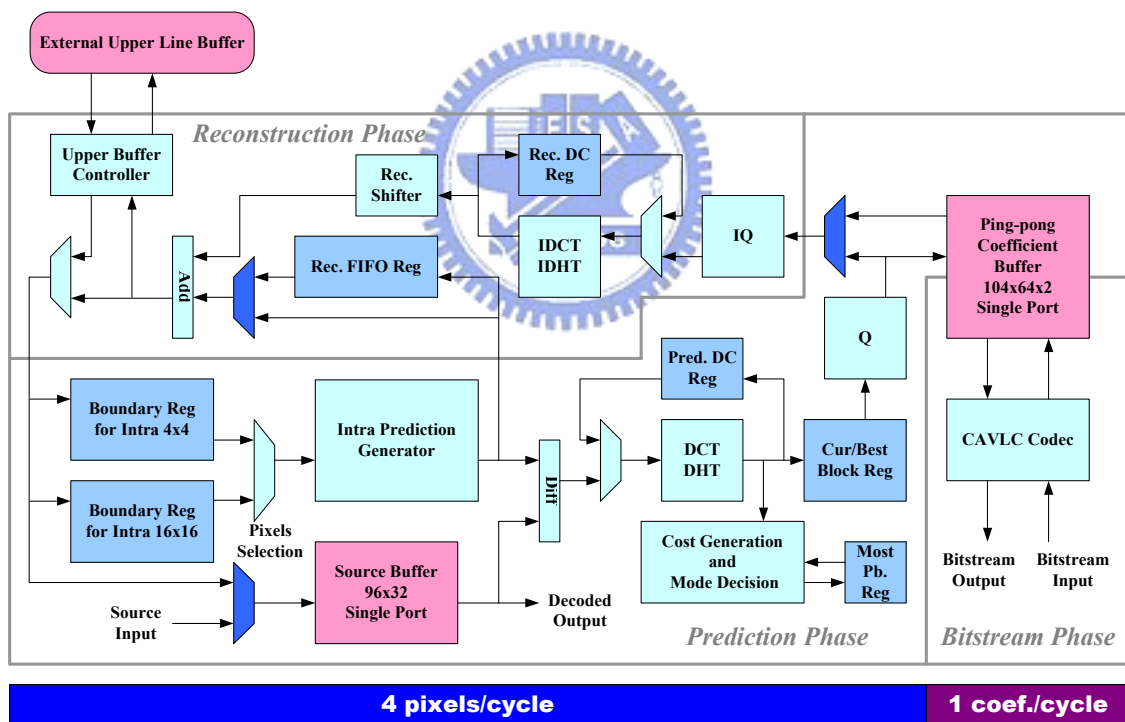


Fig. 19 Proposed architecture of baseline intra frame codec

Fig. 19 shows the architecture of the proposed codec derived from previous work [11], which is based on algorithm-level optimization and system-level pipelining mentioned

in Section 3.1 and 3.2. This design is directly corresponding to both the encoding flow in Fig. 2 and the decoding flow in Fig. 3. It can work as an intra frame encoder or a decoder with the alternative of three switch multiplexers shown in Fig. 19. The entire architecture consists of three operation phases: prediction phase, reconstruction phase, and bitstream phase.

The prediction phase is the most important part in this design. It mainly contains intra prediction generator, forward transform, cost generation and mode decision unit, quantization, and some buffers and registers. The reconstruction phase, which is used to reconstruct the decoded data, is composed of inverse transform, de-quantization, and reconstruction FIFO registers. Four-pixel parallelism is used in these two phases to achieve the required throughput. The bitstream phase is separated from previous two phases by the ping-pong buffer and uses the CAVLC codec to perform coding or decoding of bitstream with throughput of at least one coefficient per cycle. In this design, the plane prediction buffer and dual port memory are saved due to the algorithm optimizations in comparison with previous encoder-only design [13].

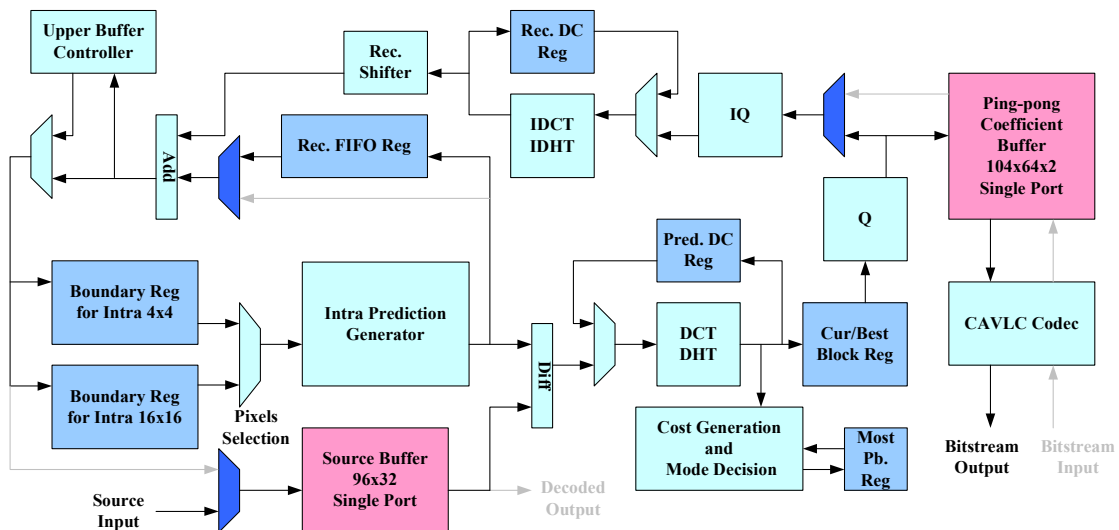


Fig. 20 Encoder dataflow of the design



decoded blocks are further sent to source memory for output and the boundary registers for next prediction. Unused components in decoder such as mode decision, forward transform, quantization, and predictor FIFO buffer are shut down to save power. Detail information for each component is discussed in the following subsections.

### **3.3.2. Schedule of Codec**

Because of variety of prediction modes, the number of process cycles for encoder is much greater than that for decoder and limits the major performance of codec. Another performance bottleneck in encoder is the reconstruction feedback loop since the next 4x4 block cannot start its computation until its boundary samples are reconstructed from previous blocks. This may result in low hardware utilization and longer latency in the prediction phase. In addition to the 4x4 block prediction, when performing 16x16 intra predictions, a macroblock-size buffer could be needed to store the processed 4x4 residual data for later mode decision, which raises the hardware cost.

To solve these issues, we propose three scheduling techniques adopted in the scheduling control unit to solve these data dependency problems and eliminate the requirement of large buffer. These three techniques are as follows:

#### **1. Insertion of the 16x16 and 8x8 predictions**

During the empty cycles waiting for reconstructed samples between two intra 4x4 blocks, the 16x16 or 8x8 intra prediction process is inserted into these bubble cycles to pre-compute their costs. Unlike the technique used in [13], the prediction generator predicts four blocks in one 16x16 mode successively instead of one block for four modes in each bubble. This helps to decrease the registers used in accumulating costs for the 16x16 prediction. After processing four blocks, it continues to process the next

4x4 prediction. Thus, utilization of components in the prediction phase is improved.

## **2. Early start of next block prediction**

Since the 4x4 blocks are processed in the Z-scan order, upper and left boundary samples might not be available at the same time for prediction purpose. To avoid this problem and pull the next block processing earlier, we rearrange the processing order of prediction modes such that prediction modes can be started as early as possible if the required data is available. For example, the vertical mode is processed before the horizontal mode since the left boundary pixels are not available. This approach can reduce the idle cycles and thus improve the throughput.

## **3. Recomputation of 16x16 and 8x8 best modes**

For the 4x4 block prediction, we use a small buffer to save the residuals of the best mode. However, when such a strategy applies to 16x16 or 8x8 predictions, a large macroblock-size buffer will be required. To solve this problem, we neglect the data generated in the prediction process and recompute them again for the best mode of 16x16 and 8x8 macroblocks after the prediction process if it is selected as the best mode. This approach may increase the total encoding cycles, but it is still in an acceptable range and can reduce the buffer cost as well.

Fig. 22 shows the pipelined schedule of this codec for processing a macroblock. Based on the above techniques, it takes at most 1,080 cycles to perform an encoding procedure for a macroblock while the best mode of luma components is selected as the 16x16 prediction. However, if the 4x4 prediction is chosen, the optional recomputation cycles between 956 and 1,024 in Fig. 22 can be eliminated, and the total cycles decrease to 1,012. Compared with the previous design [13], this work is able to save 16% of



cycle count. For decoding a macroblock, it takes 236 cycles in the average case.

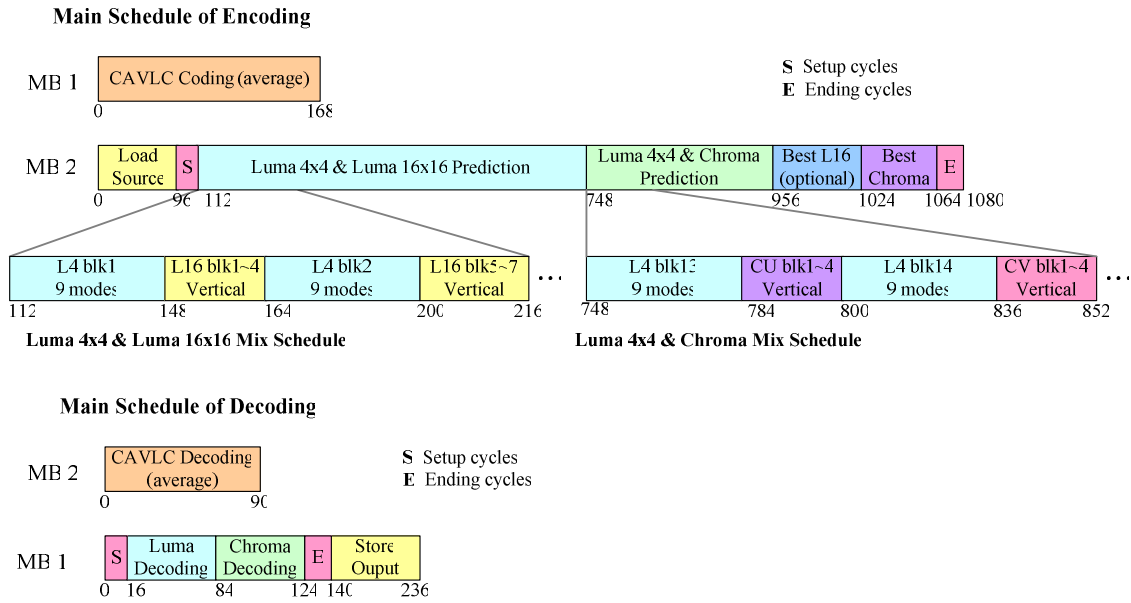


Fig. 22 Pipelined schedule for codec design

### 3.3.3. Intra Prediction Generation Unit

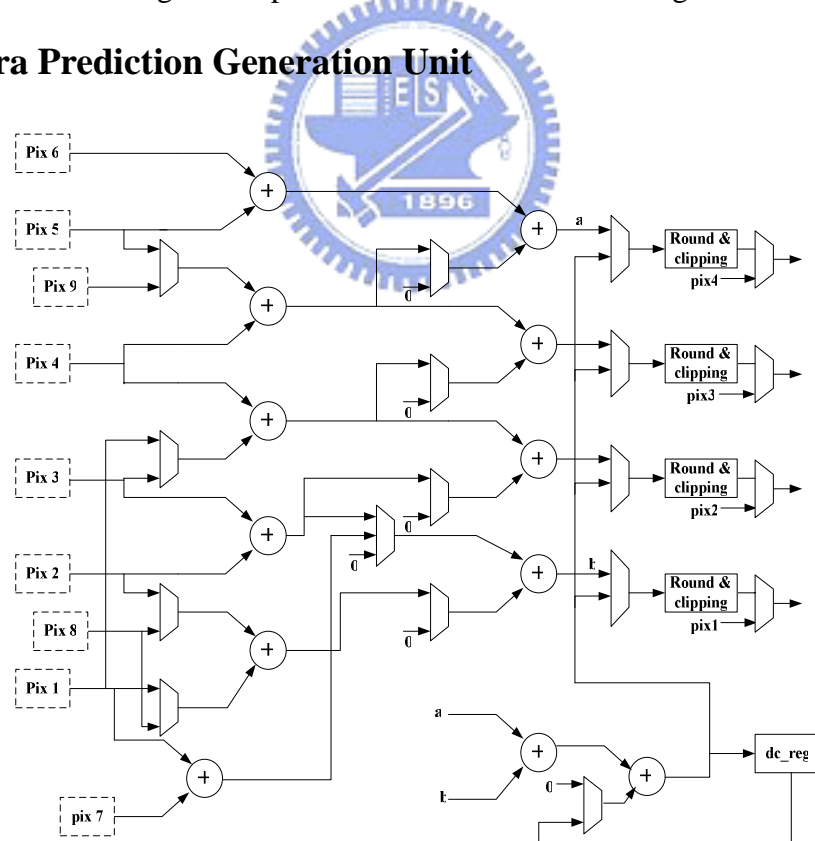


Fig. 23 Reconfigurable datapath of intra prediction generation unit

Fig. 23 shows the proposed intra prediction generation unit with the removal of plane prediction. The whole thirteen intra prediction modes are organized into four types as described in Fig. 10. Without the complex plane modes, the other modes in these types can easily share and reuse the partial sum of adjacent pixels to compute different values. To support such computation sharing, the datapath of the generator can be reconfigured to handle different modes for 4x4, 16x16, and 8x8 predictions.

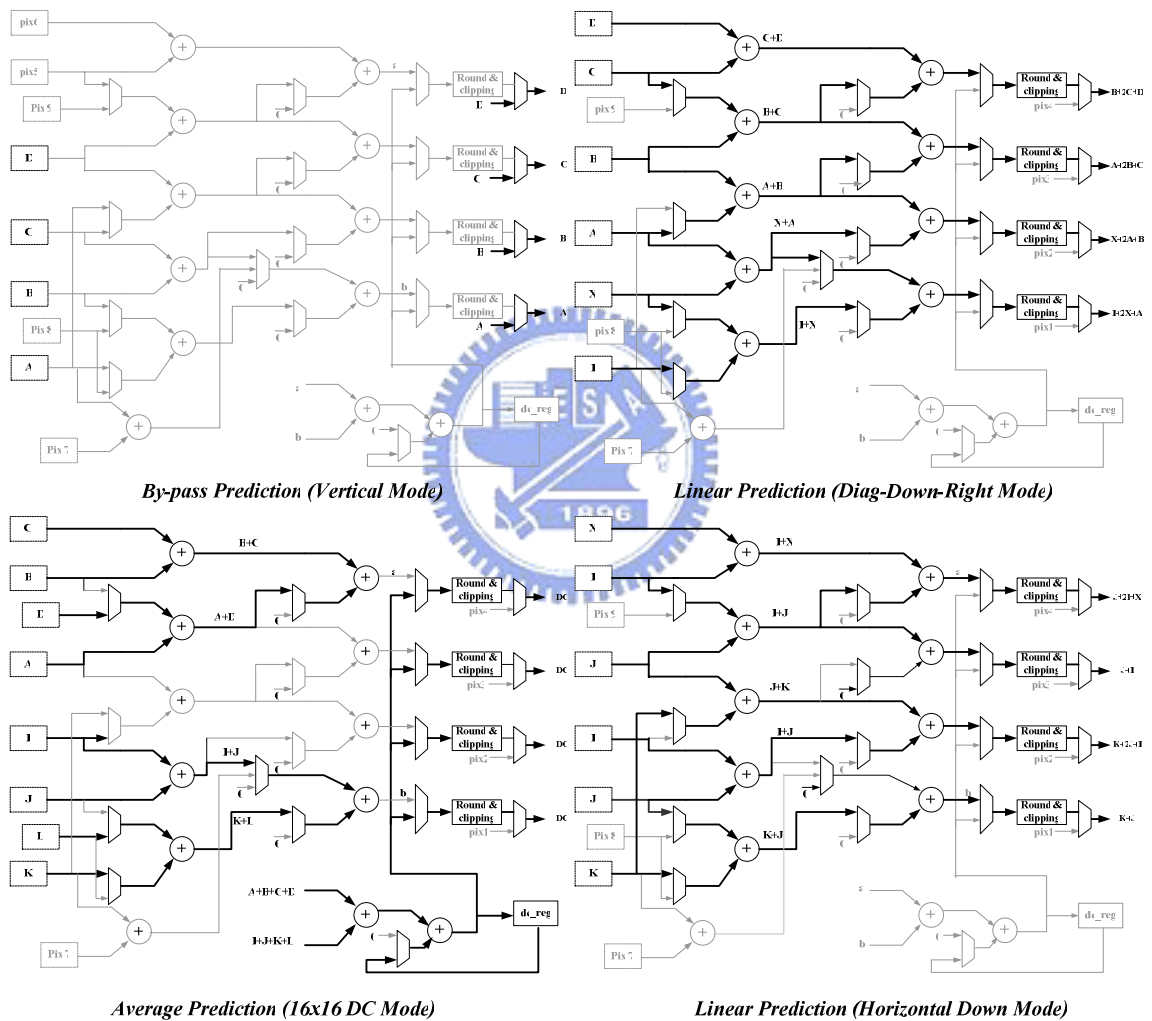


Fig. 24 Examples of operations for four intra prediction modes

The operations of this unit for four examples of prediction modes are illustrated in Fig. 24. First, the input pixels are selected from boundary register buffers that store the

neighboring pixels of previously reconstructed blocks. Then, for the bypass type, such as vertical and horizontal modes, the predictor does nothing but directly outputs the input value. For the linear type, from mode three to mode eight, desired values are obtained by reusing its partial sums. In which, the first-level adders generate values like  $(A+B+1)$ , and then the second-level adders sum up the adjacent partial sums to compute the result of  $(A+2B+C+2)$ . As to the average type, so called DC prediction, needs to sum up total eight pixels of neighboring blocks, four from upper block and four from left one, to figure out the average value each cycle. For luma 4x4 or chroma 8x8 DC modes, it takes one cycle to calculate the predicted DC value. However, four cycles are required for luma 16x16 DC prediction since up to 32 boundary pixels have to be accumulated. We use extra adders and a register to simplify the accumulation operations in the reconfigurable datapath instead of using existing adders, which saves more control and wiring circuits. After prediction, these values are handled through the difference unit to produce the residuals, which will be sent to the transform unit for further processing.

#### **3.3.4. Transform Unit**

The coefficients in transform matrices (5), (6), and (7) are even or odd symmetry at each row or column and can be easily implemented with addition and shift. The 2-D transform can also be separated into two 1-D transform with fast algorithm and butterfly architecture [16]. Since forward DCT and DHT have the same butterfly structures and will not operate at the same time in the codec, they can be merged together for area consideration. Similar architecture is applied to inverse transform. Though several transform designs have been proposed [17][18][19], this work adopts the same architecture as [20], as shown in Fig. 25, to execute integer DCT and DHT since it is

also four-pixel parallel and with lower hardware cost. In addition, two 4x4 block-size registers are located in both forward and inverse transform units to gather the DC coefficients after integer DCT and IDCT for further DHT computation of DC blocks.

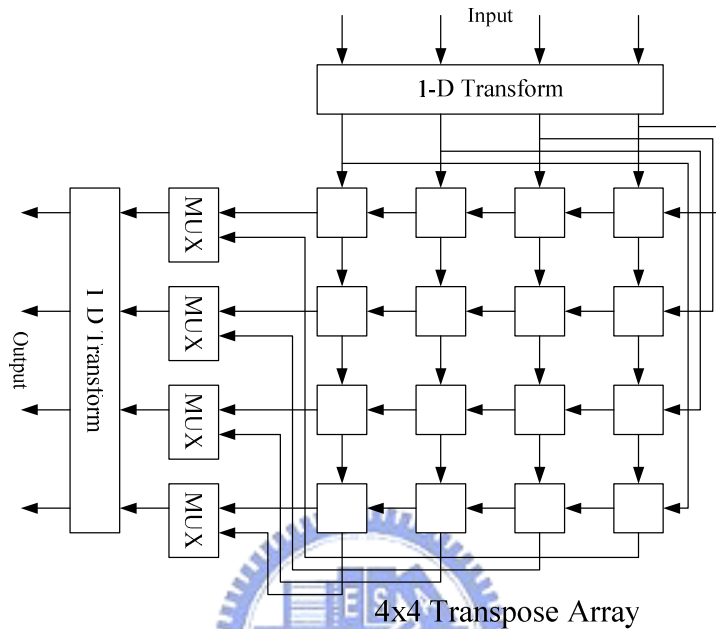


Fig. 25 Hardware architecture of transform unit [20]

### 3.3.5. Quantization and De-quantization

The quantization and de-quantization units are shown in Fig. 26, where only one of the four-parallel datapath is displayed. Constant values of quantization coefficients are all implemented by look-up tables depending on QPs from Table 2 and Table 3, as denoted by “quant\_coef,” “dequant\_coef,” “qp\_const,” “qp\_shift,” and “qp\_per” in Fig. 26. A quantized value for entropy coding is obtained through a multiplication, an addition, and a shift operation. To recover the quantized data, a multiplication followed by rounding and shift is performed. These forward and inverse processes of quantization and transform are directly matched to (6) and (7). The design also uses skill of data guarding to reduce power consumption by skipping the zero input.

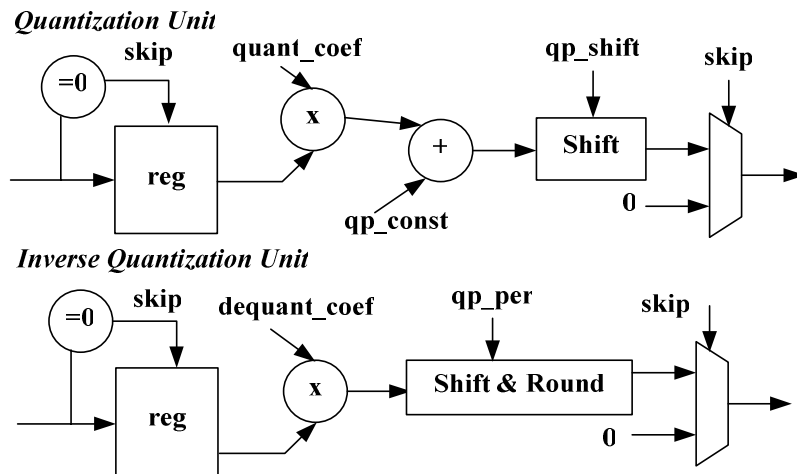


Fig. 26 Quantization and de-quantization unit

### 3.3.6. Cost Generation and Mode Decision Unit

After transform, the coefficients are sent to both cost generation unit for cost calculation and current block registers to temporarily be stored avoiding recomputation. Fig. 27 illustrates the cost generation and mode decision unit. The cost generation unit is implemented according to the enhanced SATD function in (11) that is divided in two stages: the integer transform that replaces the DHT and the extra scalar multiplication factor stage. The above replacement can eliminate the recomputation issue of transform in hardware design and also improve coding performance. The scaling integer factors are realized with a two-stage adder tree and simple shifters instead of multipliers to reduce hardware cost and critical paths.

The current cost registers are used to temporarily store the cost value for current mode. If the current mode belongs to the most probable modes, the non-zero initial cost value is given according to lambda value table instead of zero cost. The cost is then compared to the minimum cost value for 4x4 prediction or accumulated for 16x16 prediction. If a smaller cost value is detected, the minimum cost in 4x4 prediction is

replaced by the new one and coefficients in the current block registers in Fig. 19 are moved to the best block registers. This comparing and replacement procedure will be continued recursively until the best mode with minimum cost is obtained. Eventually, SATD costs of all 4x4 predictions and best one of 16x16 prediction are compared again to determine which prediction type is used in this macroblock. Similar operations are also applied to chroma components.

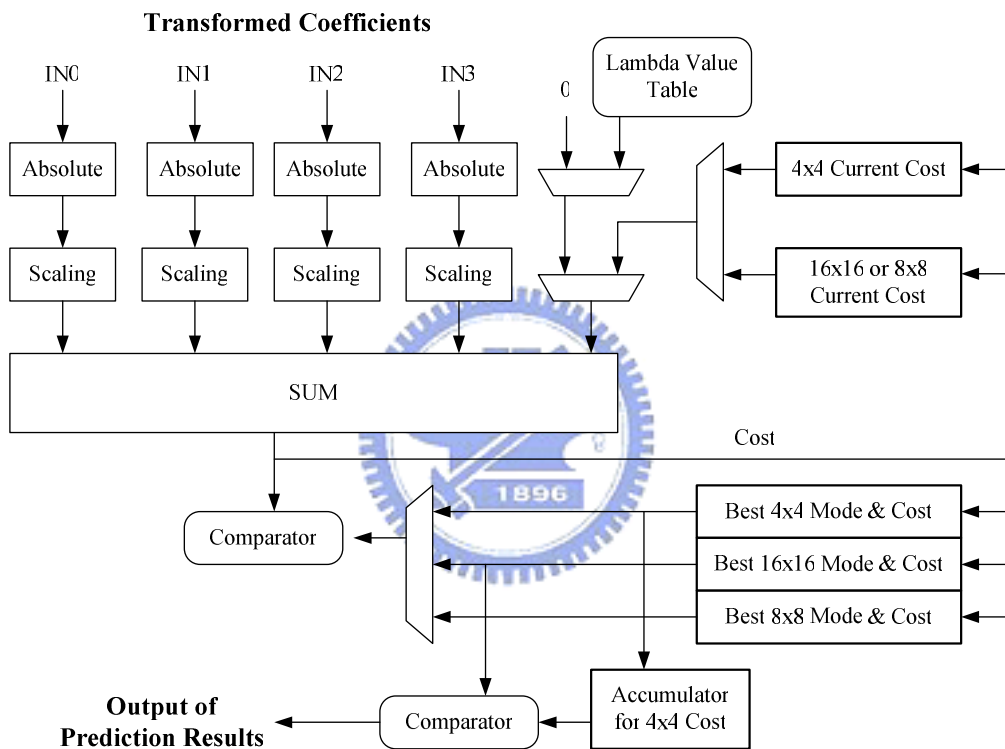


Fig. 27 Cost generation and mode decision unit

### 3.3.7. Reconstruction Path

The quantized residual values are stored in the coefficient buffer for entropy coding and also need to be reconstructed immediately since intra prediction unit requires its boundary pixels to predict successive blocks. Besides, this reconstruction process in the encoder can also be used for decoding as well. The reconstruction phase as shown in Fig.

19 consists of two paths, one for residual reconstruction and one for generation of intra prediction values. In decoder process, residuals are recovered by de-quantization, inverse transform, and shift scaling in the proper order. At the same time, the corresponding prediction modes are also obtained and added to the residuals for decoding of pixels. These intra prediction samples are immediately generated and obtained by the prediction unit in decoder but queued in the prediction FIFO in encoder due to the long latency of transform, quantization, and mode decision process. After reconstruction, some of these data are stored in the boundary buffers as the reference samples for next macroblock.

### 3.3.8. Memory Organization

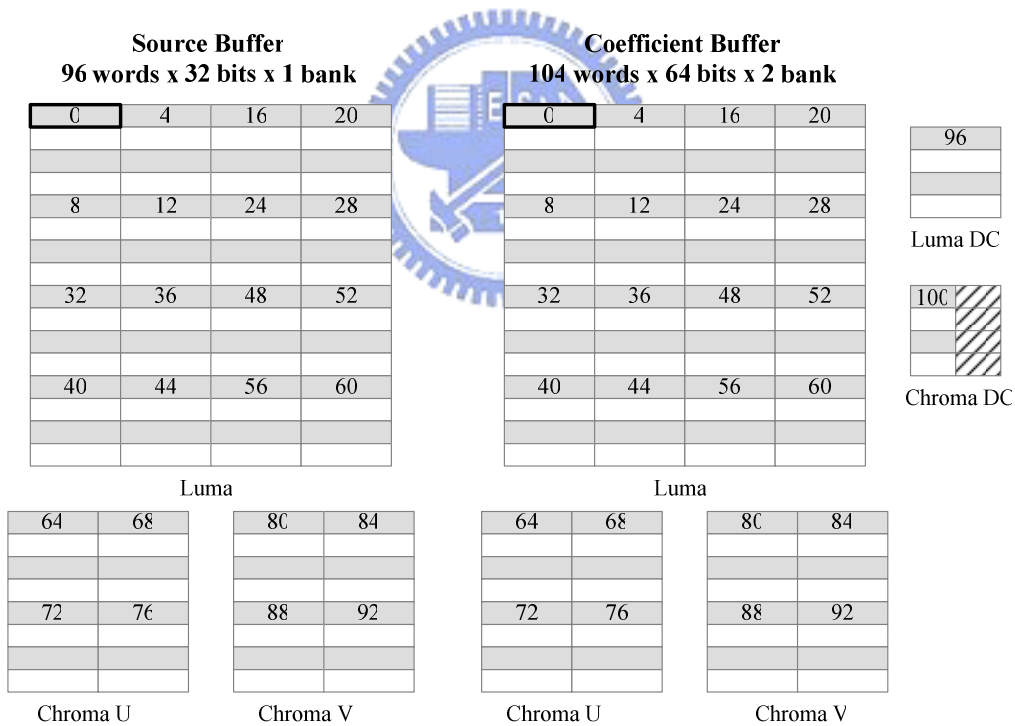


Fig. 28 Memory organization in intra codec

Only two on-chip memories are used in the proposed codec architecture, a source buffer and a coefficient buffer. The source buffer, a 96-entry 32-bit single-port SRAM,

stores four pixels per row, where 64 entries for a luma components and 32 for two chroma ones. In encoding process, this buffer stores the input source data of currently encoded macroblocks while in the decoding process, it can be used to save decoded pixels after reconstruction and then output them to the external memory at the end of decoding. The coefficient buffer adopts the ping-pong buffer architecture such that the coding loop and entropy coding stage can be pipelined to improve hardware performance. Each bank in the buffer has 104 entries with 64-bit bandwidth for 96 entries of AC coefficients and 8 entries of DC coefficients. Fig. 28 illustrates the usage of the two memory architecture. Moreover, a line buffer with frame width wide is located in the off-chip memory to store the boundary reference pixels above the current macroblock.

### 3.3.9. CAVLC Codec

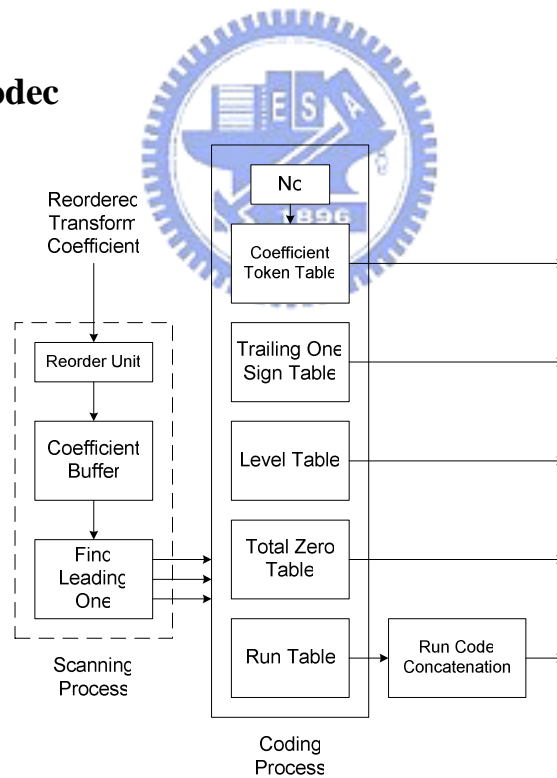


Fig. 29 Architecture for CAVLC encoder

Fig. 29 shows the hardware architecture of CAVLC encoder. The encoder can be



divided into two parts, scanning process and encoding process, which work in parallel. During encoding, the transform coefficients are first reordered in the zigzag scan order and then detected and marked as zero if it is zero. The zero coefficients coding is skipped by the leading-one detection unit and only the nonzero ones will be sent to the encoding process. Thus, the corresponding coding data is generated in parallel and sent to tables for parallel coding. This efficient zero skipped coding can speedup the process significantly, with seven cycles for one block coding in average.

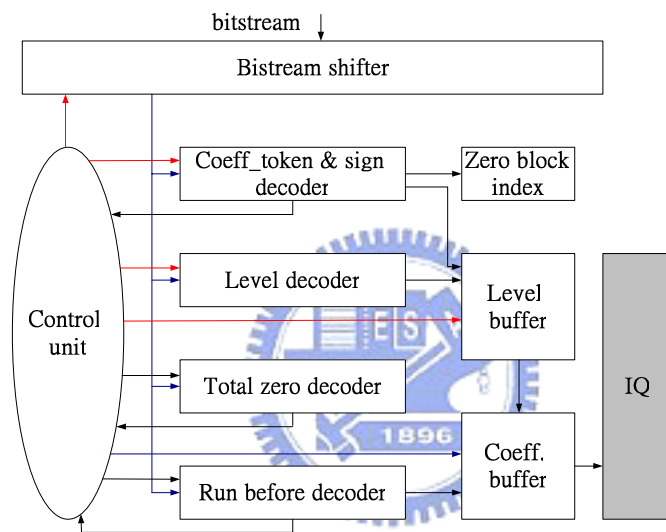


Fig. 30 Architecture for CAVLC decoder

Fig. 30 shows the hardware architecture for CAVLC decoder. This design first decodes the coefficient token and sign mark in the same cycle, and thus the level decoding can be started immediately in the next cycle. During the level decoding process, the 4x4 zero block information is recorded in the zero-block index units, and the remaining decoding circuitry is turned off for such a block. After it, the zero run decoding can be two symbols in one cycle if it is less or equal than six, a common case that is worth to be speedup. Once every run is decoded, the decoded level will be put into their corresponding position in the coefficient buffer, and no extra reordering cycle

is required to merge level and zero runs. The cycle count for one macroblock decoding is 90 in average of all sequences when QP is fixed to 28.

### 3.4. Implementation Results

#### 3.4.1. Gate-count and Layout

Table 9 List of gate count for proposed design

Component	Gate Count
Intra prediction generation	3,254
DCT/DHT with DC registers	9,657
IDCT/IDHT with DC registers	9,113
Quantization	16,493
De-quantization	4,671
Cost generation and mode decision	15,992
Reconstruction	3,804
Boundary prediction buffer	11,659
Schedule controller	1,362
CAVLC encoder	9,845
CAVLC decoder	16,421
<b>Total</b>	<b>103,057</b>

The proposed codec architecture is designed by Verilog HDL and implemented using UMC 0.18 $\mu$ m 1P6M CMOS technology. Table 9 lists the gate count for each component, where the total gate count is 103.06K. In this list, we can observe that the quantization unit occupies the largest area since it uses four multipliers in the four-pixel parallelism to perform the quantization. The four-pixel parallel design can provide smooth flow without waiting. Besides, the cost generation and mode decision unit has the second largest area due to the two block-size registers and critical timing in summery. The reconfigurable datapath and elimination of plane prediction makes our intra prediction generator only needs 3.3K gate count.

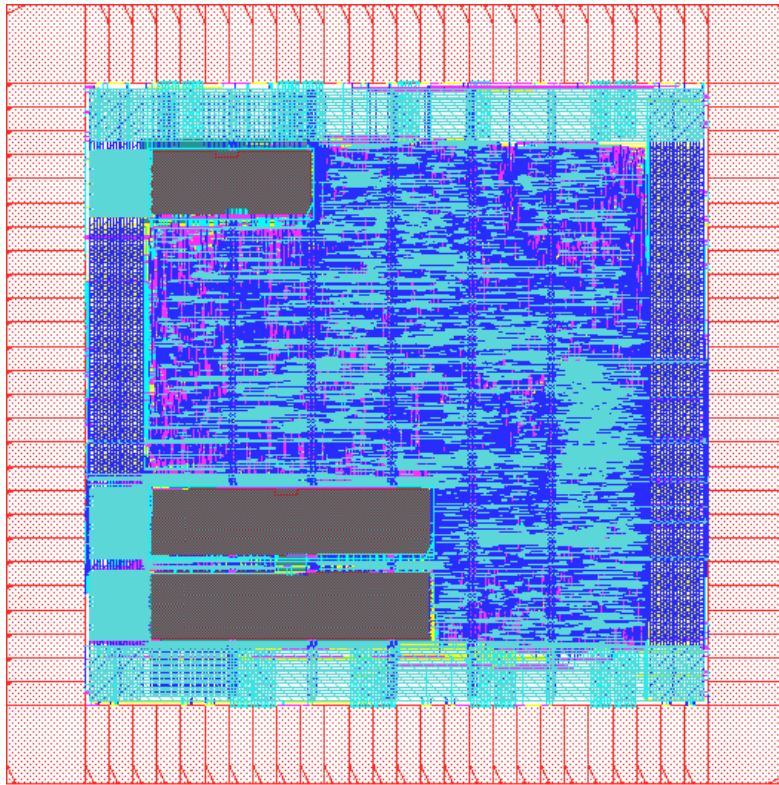


Fig. 31 Layout of the codec chip

Table 10 Information for the chip

<b>Technology</b>	UMC 0.18 $\mu\text{m}$ 1P6M CMOS
<b>Core Voltage</b>	1.8V
<b>I/O Voltage</b>	3.3V
<b>Core Size</b>	1.28x1.28 $\text{mm}^2$
<b>Package</b>	144 pin CQFP
<b>On-chip Memory</b>	Single-port 104 x 64-bit x 2 banks
	Single-port 96 x 32-bit x 1 bank

The chip layout is shown in Fig. 31 with a core size of 1.28x1.28  $\text{mm}^2$ . In this layout, one 96x32-bit SRAM is placed at the top and a pair of dual-bank 104x64-bit SRAM is implemented at the bottom of the chip. The limitation of this chip is 125MHz at the worst-case, which is larger than the required frequency, 117MHz. Thus, it can easily support 27.65M pixels/sec still image encoding and real-time video intra frame coding of high definition 720p (1280x720 4:2:0 at 30fps) video application when clocked at

117MHz. Besides, it can also support the same decoding throughput with only frequency of 25.5MHz, and larger size for high definition 1080p with frequency of 58MHz. Table 10 lists the chip information.

### 3.4.2. Comparison

Since no intra codec has been published, we compare our design with other encoder and decoder design as shown in Table 11 . Previous encoder design [13] can only support SD size (720x480 4:2:0 at 30 fps) at frequency of 54MHz and has a total gate count of 84K implemented in TSMC 0.25 $\mu$ m 1P5M CMOS technology. However, it uses two single-port and one dual-port memories in their architecture and results in a larger core size. In addition, it cannot support the decoding process and needs extra area cost for plane mode prediction. The proposed design can support both codec and HD size processing with slightly larger area and lower operating frequency. The extra area cost is roughly the area cost of CAVLC decoder. The other design [21] is a pure baseline decoder implementation that supports HD 1080p size decoding with frequency of 100MHz. In comparison, our design needs lower operating frequency for the same decoding capability.

Table 11 Comparison among [13], [21], and this work

Design Feature	This Work	[13]	[21]
Max operation freq.	125MHz	55MHz	100MHz
System pipeline	MB-based	MB-based	Block-based
Pixel parallelism	4 pixels	4 pixels	4 pixels
CMOS technology	UMC 0.18 $\mu$ m	TSMC0.25 $\mu$ m	UMC 0.18 $\mu$ m
Gate count	103K	85K	450K
Chip core size	1.28x1.28 mm <sup>2</sup>	1.86x1.86 mm <sup>2</sup>	N/A
On-chip memory usage (for 720p)	Single 96x32(x1)	Single 96x32(x2)	Single 96x32(x2)
	Single 104x64(x2)	Single 64x32(x1)	Single 80x32(x1)

		Dual 96x16 (x4)	
Off-chip memory usage (for 720p)	Single 80x32 SDRAM	Single 80x32 SDRAM	N/A
<b>Encoder</b>	Supported	Supported	Unsupported
Max target size	HD 1280x720	HD 720x480	N/A
Freq. for HD 720p	117MHz	N/A	N/A
Freq. for SD	43MHz	54MHz	N/A
Cost function	Enhanced DCT-based SATD	DCT-based SATD	N/A
Plane prediction buffer	No	Yes	N/A
Plane mode prediction	No	Yes	N/A
DC values forwarding	Yes	No	N/A
Coefficient reg. size	4x4 block	16x16 MB	N/A
Processing cycles/MB	< 1080 cycles	< 1300 cycles	N/A
<b>Decoder</b>	Supported	Unsupported	Supported
Max target size	HD 1920x1080	N/A	HD 1920x1080
Freq. for HD 1080p	58MHz	N/A	100MHz
Freq. for HD 720p	25MHz	N/A	50MHz
Freq. for SD	10MHz	N/A	20MHz
Inter prediction	No	N/A	Yes
Processing cycles/MB	= 236 cycles	N/A	Unknown

## Chapter 4

# Fast H.264/AVC Intra Frame Encoder

In Chapter 3 an H.264/AVC baseline profile intra frame codec is proposed for high definition video application. This codec design can support both encoder process for HD size 720p and decoder process for HD size 1080p at the working frequency lower than the previous designs. Though this design is quite suitable for video application products such as digital camera or digital video recorder, it still lacks the consideration for power consumption. With the popularity and demand of the portable products, the power issue plays an important role in the current SOC design.

To solve this problem and focus on the power-saving techniques, a new H.264/AVC baseline profile intra encoder with variable-pixel parallel architecture and modified three-step fast prediction algorithm is presented in this chapter. The variable-pixel parallel datapath can save almost half of the prediction cycles efficiently, and the fast algorithm is able to speedup the prediction flow of 4x4 modes with only negligible quality degradation. In comparison with previous codec design, this work has the same ability to support HD size 720p encoding process but with nearly half of frequency requirement, only 61MHz. With the modified three-step fast prediction algorithm, eight-pixel parallel prediction engine, and enhanced designs, both cycle latency and hardware area can be reduced for lower power consumption.

## 4.1. Fast Algorithm for Intra Prediction

### 4.1.1. Survey of Fast Algorithm

In the intra coding of H.264/AVC, intra prediction and SATD cost function for mode decision take almost 77% of computation in all functions [13]. This result is reasonable since for a 4x4 block, there are thirteen modes for luma prediction and four for chroma prediction. Thus, the SATD function has to be applied to a luma block for thirteen times but to its best mode only once. If some unnecessary prediction modes can be eliminated by a fast algorithm, we can reduce the computation of intra prediction and its related SATD transform.

Various fast algorithms for intra predictions are published to decrease prediction modes efficiently with acceptable performance loss. For example, some algorithms define thresholds for RDO cost [22] or for SAD cost [23] to select the modes to be skipped. If the cost of currently predicted mode is smaller than the pre-defined threshold, the other modes are skipped. The other algorithms use special methods to predict the appropriate modes such as macroblock properties prediction [24], edge map and local edge direction histogram [25], and feature-based mode filtering [26]. They take advantage of the correlation of intra-block or inter-block pixels to speedup the decision process. Further solution to reduce computation by combining intra prediction with transform [27] is also proposed. However, these researches are mostly developed in only algorithm level with software optimization but not for hardware design purpose. As a result, most of the algorithms are quite computationally intensive, hard to be implemented in hardware, and memory-demanding.

A suitable choice for both simple hardware addition and lowly extra computation is using the fast three step intra prediction algorithm [28] as shown in Fig. 32. This algorithm uses the existing SATD results to predict a best mode for a 4x4 block through a three-step flow, which statically decreases the prediction modes from nine to six. Such

fast algorithm focusing on reduction of 4x4 prediction modes benefits to effectively decrease total computation cycles since number of operations for the 4x4 prediction is significantly more than those for 16x16 prediction according to the complexity analysis [29]. The simulation results in [28] show that it can save about 33% of computation for the intra prediction and related transform with a bitrate loss of 1%. Besides, this algorithm is suitable for the hardware implementation with little comparison circuit.

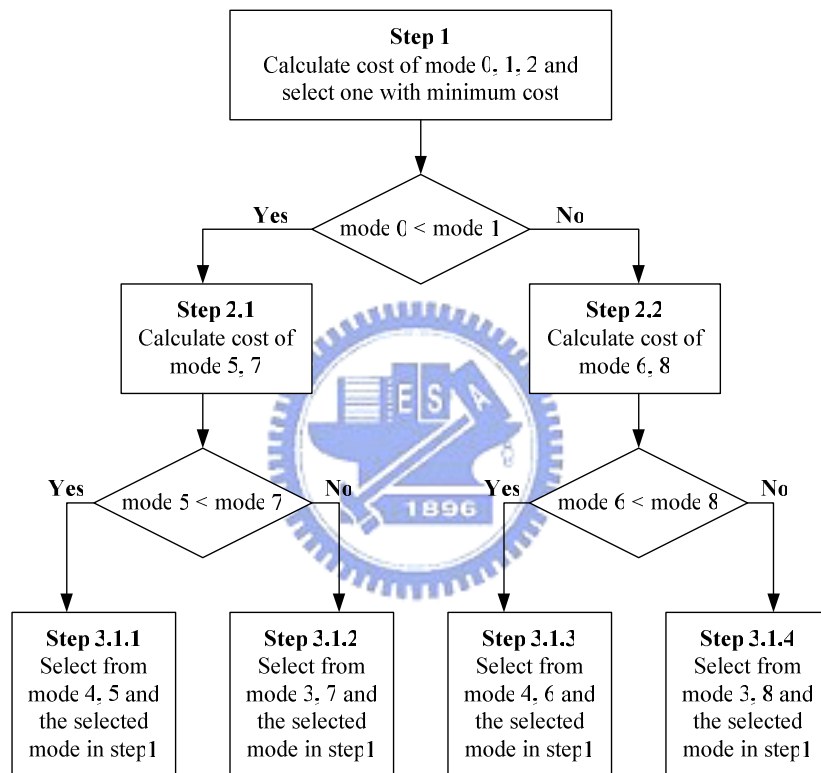


Fig. 32 Decision flow of fast three-step algorithm for intra prediction

#### 4.1.2. Modified Fast Algorithm for Intra Prediction

Through the three-step algorithm is more suitable for hardware design than the other software-based ones, it still have much room for improvement in practice. To fit the pipeline architecture and schedule in our encoder design, the order of the decision flow should be properly modified.



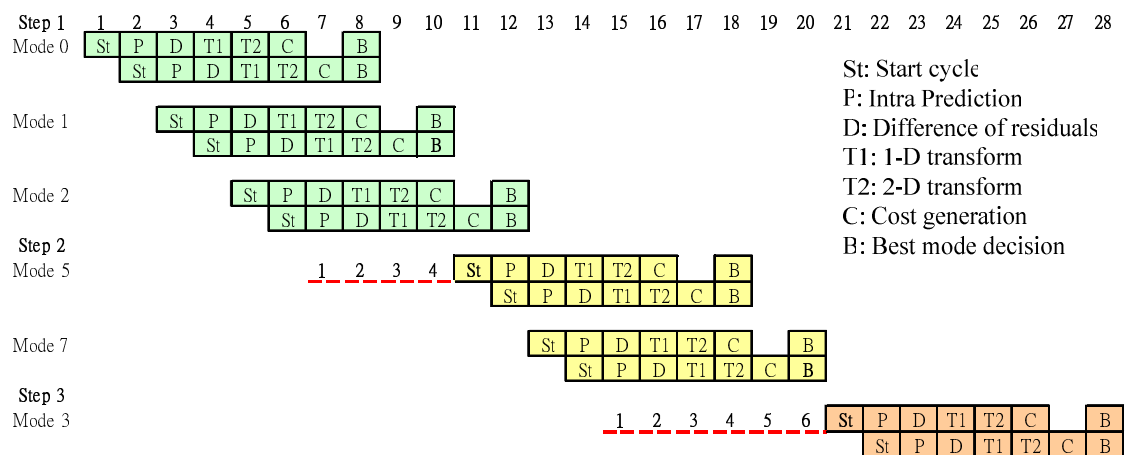


Fig. 33 Fast three-step algorithm in pipeline structure

Fig. 33 illustrates an example to directly apply the three-step algorithm used in the eight-pixel parallel pipeline architecture modified from previous work of codec. In the pipeline stage diagram, each block takes 8 cycles latency to complete a prediction process including intra prediction, SATD function, and mode decision, and the first step in Fig. 32 will take 12 cycles for three modes. The second step can be immediately executed after the 10th cycle in the first step since the comparison results of mode 0 and mode 1 is obtained. However, this scheduling leads to four cycle bubbles. The same situation also exists at the transition between the second step and the third step that generates six cycles latency. Therefore, total 28 cycles are required to predict a block with the original fast algorithm.

A solution to conceal the bubble cycles is to adjust the order of prediction modes in the scheduling. Since the third step in Fig. 32 has to predict either mode 3 or mode 4 no matter which branch is chosen, we can move these two modes to the second step and fill the transition bubbles. Fig. 34 illustrates a new decision flow for the modified fast algorithm with only one decision. Though the prediction modes increase from six to seven, the total cycles to predict a block are reduced to 20 and no bubble cycle exists.

Thus, pipeline operations can be executed successively without cycle waste. Fig. 35 shows the modified algorithm in the eight-pixel parallel pipeline architecture.

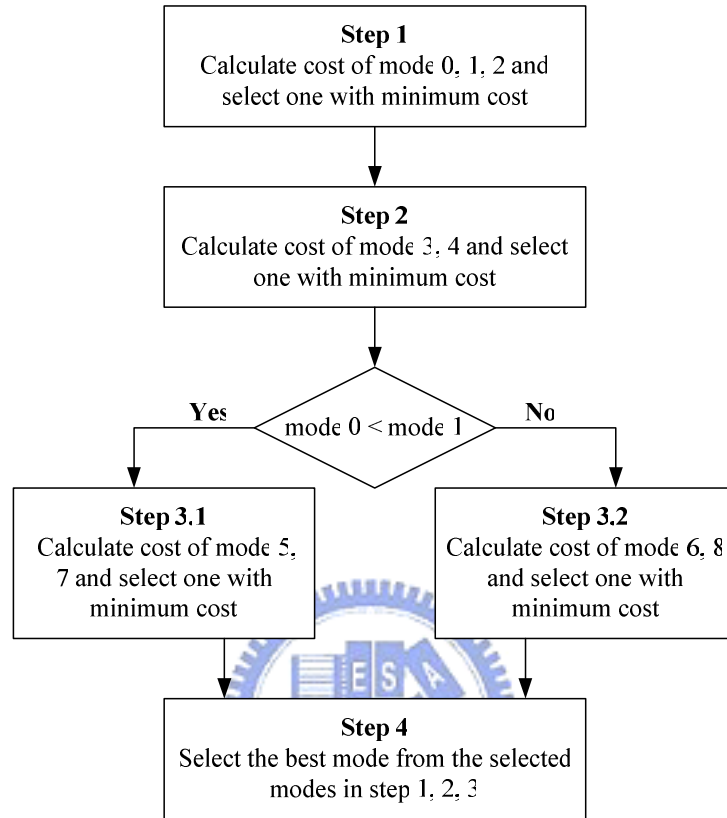


Fig. 34 Decision flow of modified three-step algorithm for intra prediction

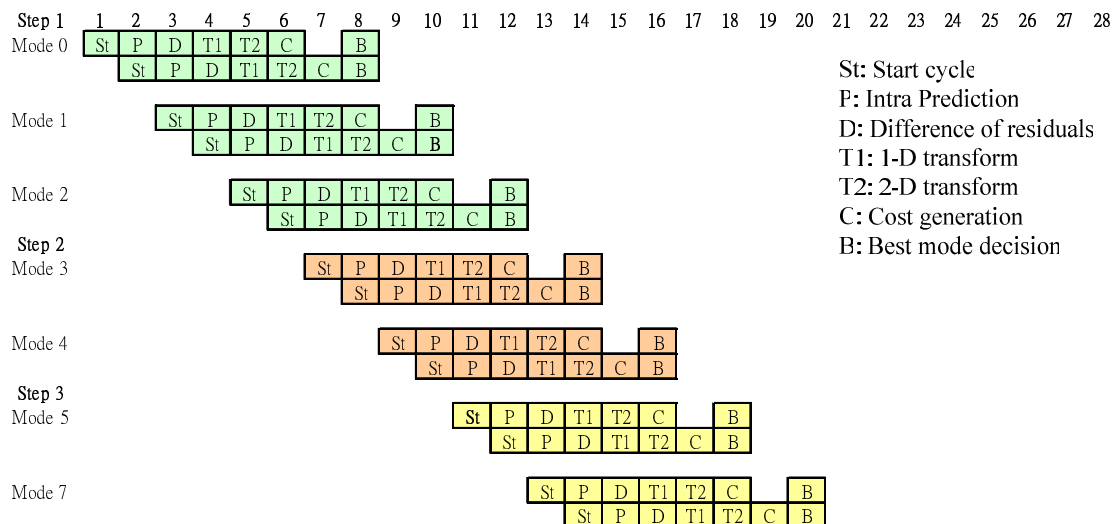


Fig. 35 Modified fast three-step algorithm in pipeline structure

### 4.1.3. Simulation Results

Table 12 Comparison among original [10], modified algorithm, and proposed fast algorithm combined of three techniques for 300 Intra frames

Sequence	QP	JM 8.6				Modified 3-step Algorithm				Combined Algorithm			
		SNR Y	SNR U	SNR V	Bit-rate	SNR Y	SNR U	SNR V	Bit-rate	SNR Y	SNR U	SNR V	Bit-rate
Stefan	16	46.38	47.27	47.43	10537.41	+0.00	+0.00	+0.00	+0.26%	+0.05	+0.00	+0.00	+0.19%
	20	42.96	44.43	44.51	8143.18	+0.01	+0.00	+0.00	+0.33%	+0.02	-0.03	-0.03	+0.24%
	24	39.63	41.63	41.65	6189.86	+0.00	+0.00	+0.00	+0.42%	-0.04	-0.14	-0.12	+0.32%
	28	36.41	38.95	38.96	4585.29	+0.00	+0.00	+0.00	+0.55%	-0.05	-0.23	-0.25	+0.43%
	32	33.05	37.12	37.08	3246.41	-0.01	+0.00	+0.00	+0.67%	-0.09	-0.42	-0.45	+0.56%
	36	29.96	35.15	35.07	2218.46	-0.01	+0.00	+0.00	+0.79%	-0.12	-0.53	-0.55	+0.74%
Mobile	16	45.93	46.25	46.29	15361.27	+0.00	+0.00	+0.00	+0.32%	+0.04	+0.01	+0.01	+0.28%
	20	42.14	42.87	42.89	12233.46	+0.00	+0.00	+0.00	+0.38%	+0.03	-0.03	-0.03	+0.33%
	24	38.49	39.72	39.68	9487.00	-0.01	+0.00	+0.00	+0.46%	-0.01	-0.11	-0.10	+0.42%
	28	35.04	36.88	36.76	7179.38	+0.00	+0.00	+0.00	+0.59%	+0.00	-0.16	-0.18	+0.54%
	32	31.50	34.89	34.67	5200.07	+0.00	+0.00	+0.00	+0.75%	-0.02	-0.24	-0.23	+0.75%
	36	28.28	32.87	32.60	3572.40	-0.01	+0.00	+0.00	+0.89%	-0.06	-0.31	-0.22	+0.96%
Paris	16	46.16	47.35	47.63	10114.93	+0.00	+0.00	+0.00	+0.38%	+0.06	-0.05	-0.04	+0.42%
	20	42.81	44.69	44.88	7662.99	+0.00	+0.00	+0.00	+0.50%	-0.01	-0.20	-0.15	+0.55%
	24	39.59	41.97	42.12	5742.80	+0.00	+0.00	+0.00	+0.59%	-0.10	-0.36	-0.28	+0.71%
	28	36.49	39.40	39.54	4235.43	+0.00	+0.00	+0.00	+0.71%	-0.11	-0.52	-0.36	+0.89%
	32	33.33	37.51	37.73	3005.78	-0.01	+0.00	+0.00	+0.94%	-0.21	-0.65	-0.52	+1.12%
	36	30.36	35.58	35.78	2055.30	+0.00	+0.00	+0.00	+0.99%	-0.27	-0.67	-0.56	+1.19%
Akiyo	16	47.34	48.46	49.40	4159.54	+0.00	+0.00	+0.00	+0.74%	+0.09	-0.06	-0.12	+1.13%
	20	44.89	46.88	47.95	2777.52	-0.01	+0.00	+0.00	+0.80%	-0.05	-0.30	-0.38	+1.10%
	24	42.65	44.62	46.03	1941.72	+0.00	+0.00	+0.00	+1.49%	-0.25	-0.37	-0.64	+2.16%
	28	40.33	42.52	43.93	1370.30	-0.02	+0.00	+0.00	+1.51%	-0.37	-0.63	-0.85	+2.11%
	32	37.77	40.82	42.54	963.42	-0.03	+0.00	+0.00	+1.30%	-0.62	-0.69	-1.09	+1.98%
	36	35.28	38.80	40.69	672.96	-0.04	+0.00	+0.00	+1.32%	-0.75	-0.64	-0.98	+1.56%
Foreman	16	46.26	47.56	48.57	7665.68	+0.00	+0.00	+0.00	+0.40%	+0.07	+0.03	+0.00	+0.41%
	20	42.90	45.17	46.83	5394.72	+0.00	+0.00	+0.00	+0.55%	+0.08	-0.08	-0.21	+0.60%
	24	39.95	42.87	44.78	3678.28	-0.01	+0.00	+0.00	+0.69%	-0.04	-0.25	-0.35	+0.82%
	28	37.26	40.91	42.79	2467.35	-0.01	+0.00	+0.00	+0.80%	-0.09	-0.34	-0.46	+0.95%
	32	34.61	39.80	41.32	1598.24	-0.01	+0.00	+0.00	+0.83%	-0.23	-0.46	-0.60	+1.20%
	36	32.24	38.61	39.81	1022.62	-0.01	+0.00	+0.00	+0.74%	-0.35	-0.45	-0.57	+1.26%
Coastguard	16	45.88	48.20	49.05	9454.85	+0.00	+0.00	+0.00	+0.21%	+0.04	+0.06	+0.05	+0.06%
	20	42.13	46.38	47.64	6969.04	+0.00	+0.00	+0.00	+0.29%	+0.09	-0.05	-0.11	+0.12%
	24	38.74	44.64	46.13	4959.03	+0.00	+0.00	+0.00	+0.41%	+0.06	-0.24	-0.13	+0.28%
	28	35.63	43.08	44.72	3437.66	-0.01	+0.00	+0.00	+0.52%	+0.11	-0.35	-0.18	+0.41%
	32	32.74	41.96	43.72	2236.41	+0.00	+0.00	+0.00	+0.58%	-0.01	-0.44	-0.23	+0.49%
	36	30.24	40.82	42.72	1418.33	-0.01	+0.00	+0.00	+0.59%	-0.16	-0.39	-0.14	+0.59%

Table 12 shows the comparison results between the original mode decision method and the modified ones for six common sequences. It can be observed that the usage of the modified three-step algorithm only increases 0.68% of bitrate in average, compared to the results from [10]. Though the bitrate is increased, the 4x4 prediction modes can

be reduced from nine to seven with 23% of computation saving and the quality of PSNR is almost unchanged. After we combine the modified fast three-step algorithm with previously mentioned enhanced SATD cost function and plane mode removal in Section 3.1, the average bitrate increase for six sequences is slightly changed to 0.77%, which is still better than the original algorithm [28] and other fast algorithms.

The RD curve diagrams of simulation results for [10] and the combined fast algorithm are shown from Fig. 36 to Fig. 41. Except “Akiyo”, the QP range is restricted from 20 to 32 for these diagrams. Though there is tiny performance loss in some sequences like “Akiyo” and “Paris” due to their low bitrate when QPs are large, however in most cases, the combined algorithm still makes the coding performance much similar or even better in low QP range.

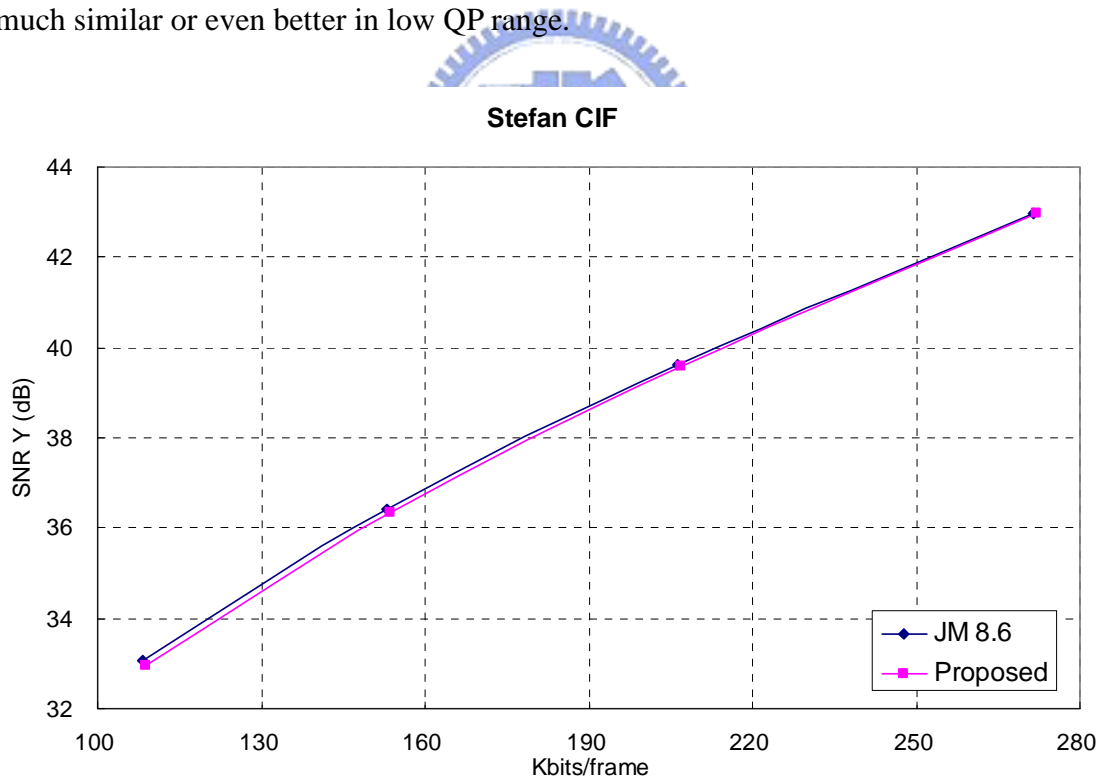


Fig. 36 RD curves of [10] and proposed fast algorithm for sequence “Stefan”

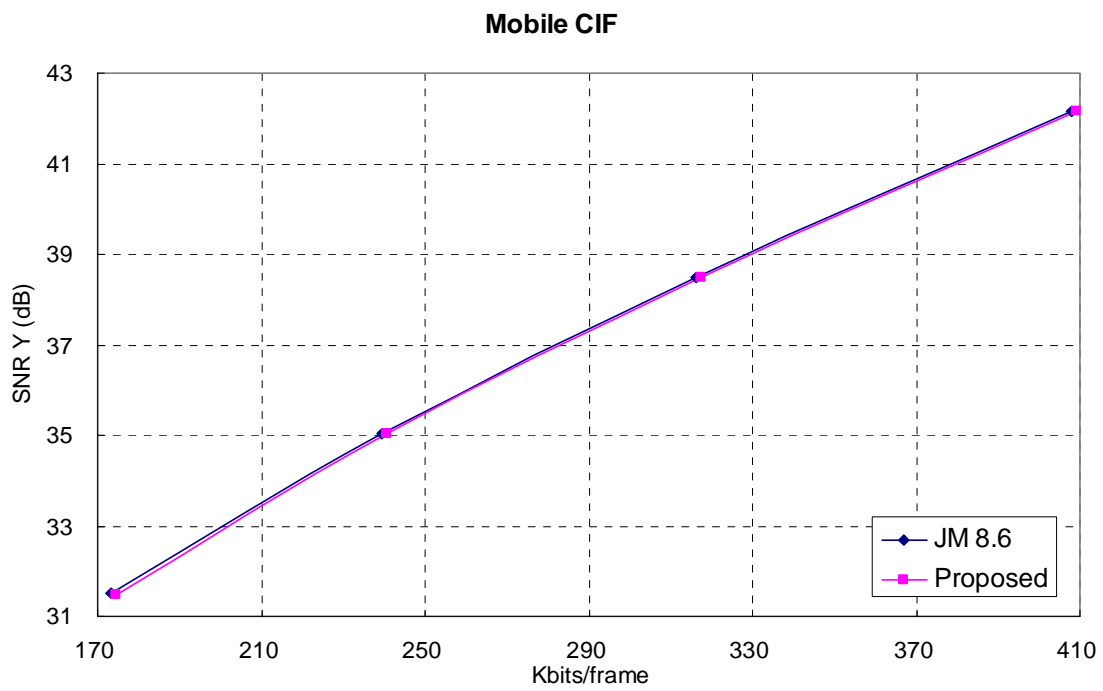


Fig. 37 RD curves of [10] and proposed fast algorithm for sequence “Mobile”

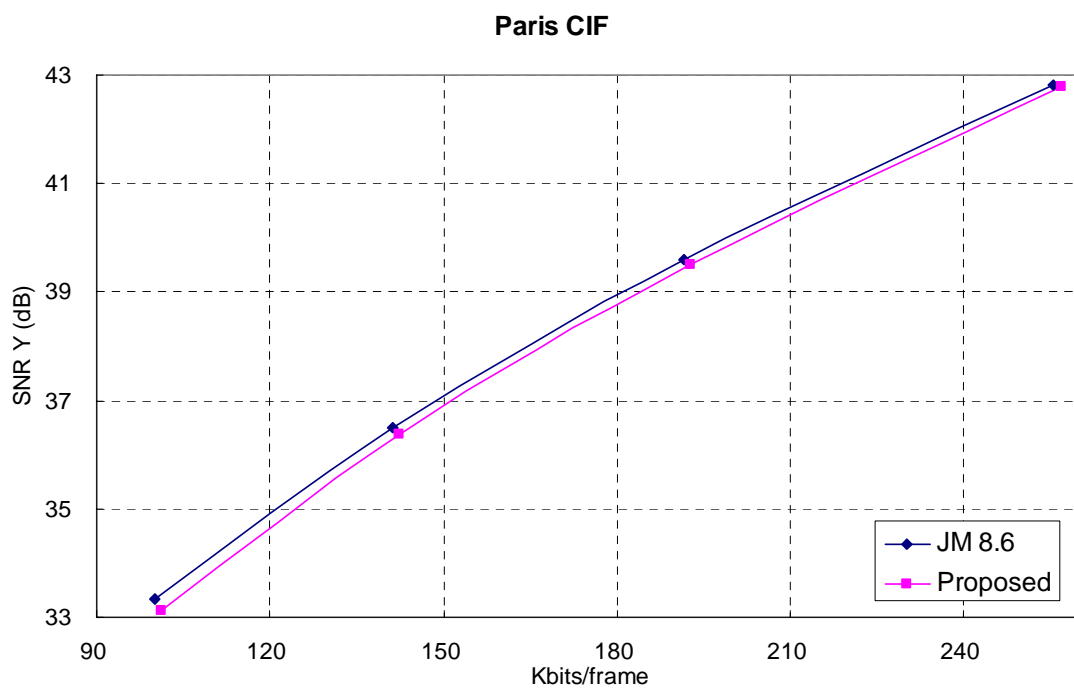


Fig. 38 RD curves of [10] and proposed fast algorithm for sequence “Paris”

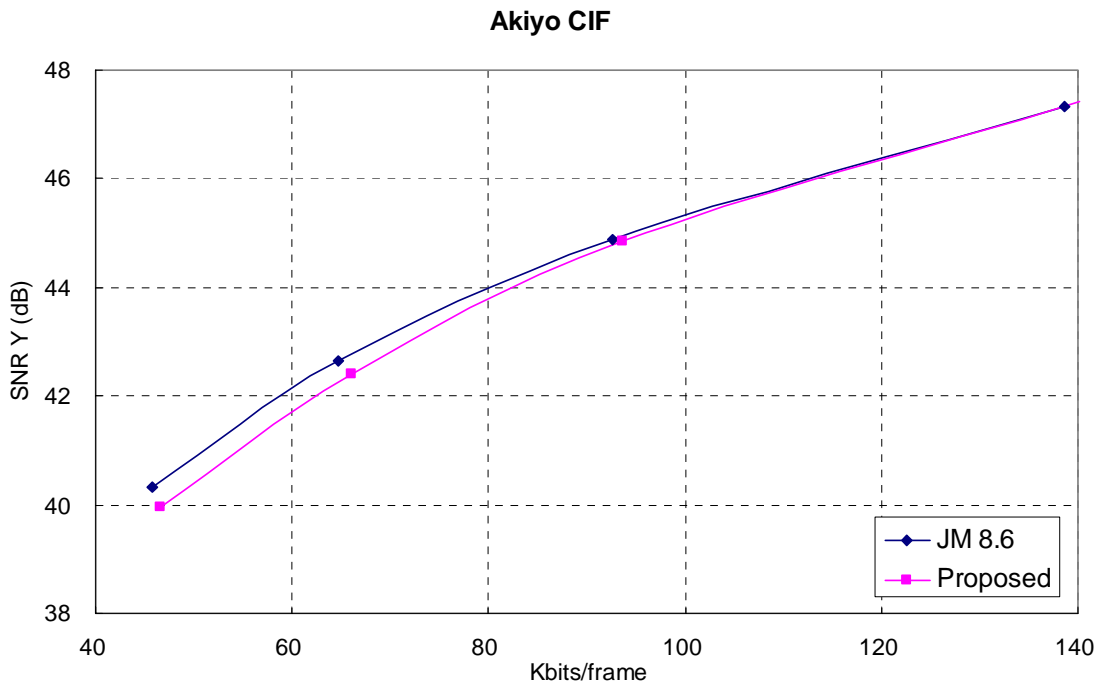


Fig. 39 RD curves of [10] and proposed fast algorithm for sequence “Akiyo”

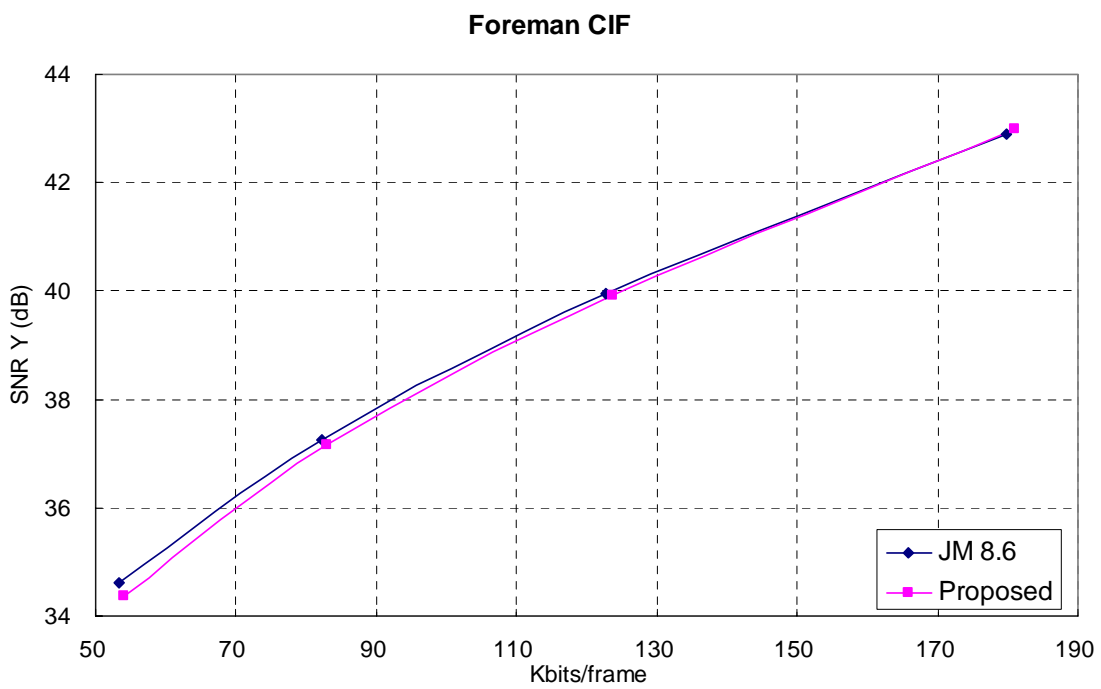


Fig. 40 RD curves of [10] and proposed fast algorithm for sequence “Foreman”

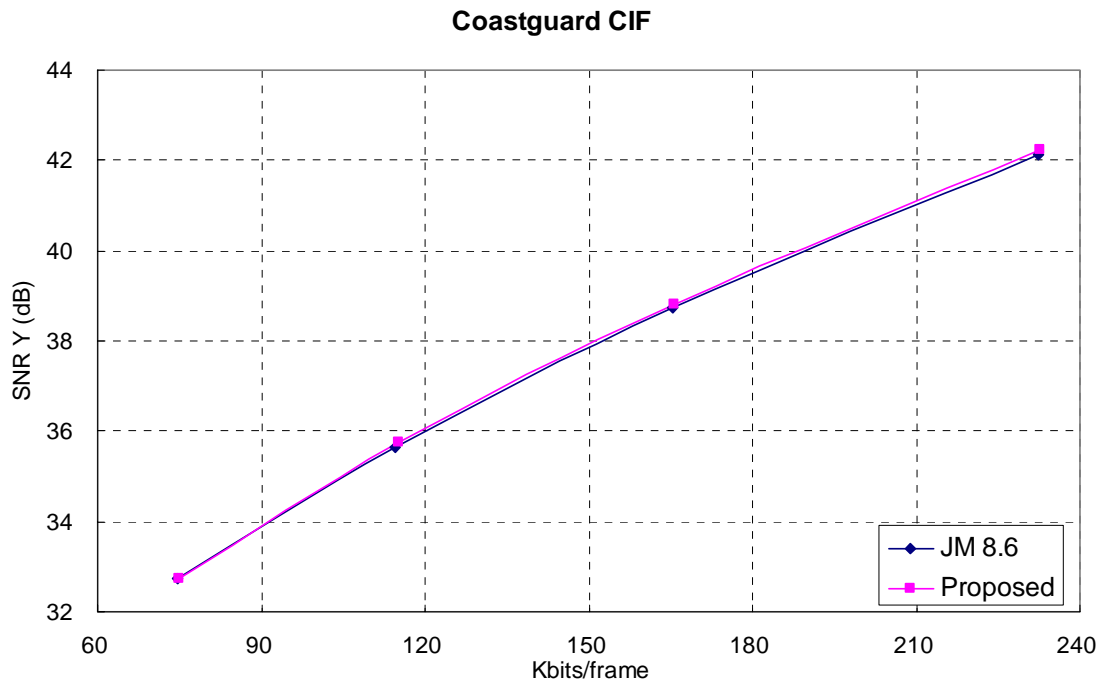


Fig. 41 RD curves of [10] and proposed fast algorithm for sequence “Coastguard”

## 4.2. Architecture Design of Fast Intra Encoder

### 4.2.1. Overall Architecture

The proposed intra frame encoder design with the modified fast algorithm and variable-pixel parallel architecture is shown in Fig. 42. This design is extensively modified from previous work in Chapter 3 and mainly partitioned into four phases, prediction phase, reconstruction phase, quantization phase, and bitstream phase. The major changes from previous design are eight-pixel instead of four-pixel parallelism in the prediction phase. The quantization and reconstruction phase are kept unchanged, four-pixel parallelism, and thus partitioned from the prediction phase.

The eight-pixel parallel prediction phase significantly improves the throughput and reduces the cycles for the computationally critical intra prediction generator by half.

This decision will not increase the area cost by one time since we only add one more intra prediction engine, two 1-D four-point transform, and a few small buffers. The total architecture includes a pair of boundary buffer, two intra prediction engines, an eight-input 4x4 transform, a cost generator with feedback signals for fast mode decision, and some registers. Since only blocks with best modes are allowed to pass through the quantization phase and reconstruction phase, these two phases adopt four-pixel parallel architecture to save area. To allow smooth data flow between different data parallelism, we use the current block and best block registers in the quantization phase and the FIFO registers to buffer the data. The bitstream phase, including CAVLC encoder, is similar to [11] and slightly modified to fit the coding bitstream with at least one coefficient per cycle.

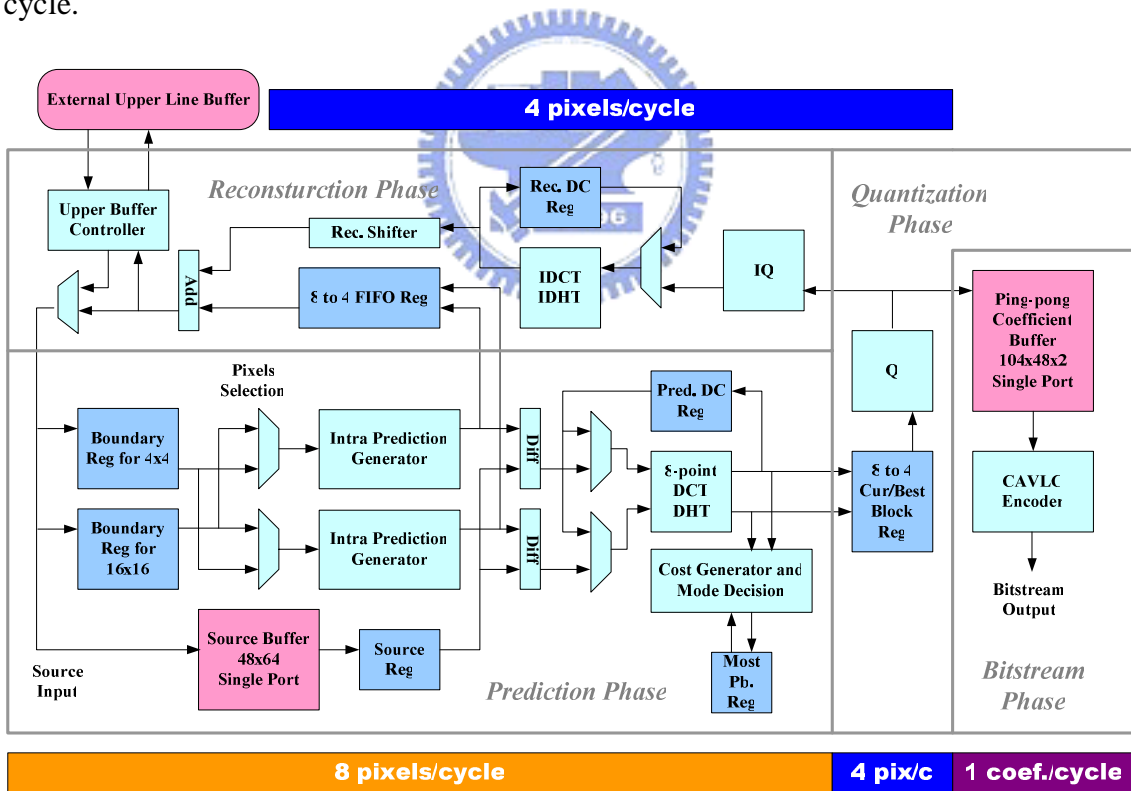


Fig. 42 Proposed architecture of encoder with fast algorithm



### 4.2.2. Scheduling of Encoder

In addition to the feedback loop in the encoder, the newly adopted variable-pixel parallel architecture also restricts the performance in the design. For example, though a block-size buffer is placed at the interface between four-pixel parallel quantization phase and eight-pixel parallel prediction phase, the recomputed coefficients for best luma 16x16 and chroma modes cannot be passed through successively because of the bottleneck of different parallelism and thus will be blocked. This situation will result in a larger buffer to store temporarily blocked data or low utilization with empty cycles in the prediction phase.

An efficient solution to this problem without utilization loss is to use the interlaced pipelined schedule for best mode as shown in Fig. 43 and Fig. 44. We interleave and insert the successively recomputed best modes for luma 16x16 and chroma components into the normal prediction modes that do not need the four-pixel parallel datapath. This can improve the high utilization in the prediction phase without wasted cycles and maintain the data continuity in the quantization phase as well. However, a few empty cycles are still needed for best chroma mode only scheduling as shown in Fig. 43. Besides, to solve the frequent transient states caused by interlaced schedule and simplify the control unit, a counter-based controller is used instead of the traditional finite state machine controller. With the interlaced method and the previous three techniques: insertion of 16x16 and 8x8 prediction, early start of next prediction, and recomputation for best mode, the total cycle count for encoding a macroblock can be reduced to 522 or 560, about only 52% relative to that in previous design [11].

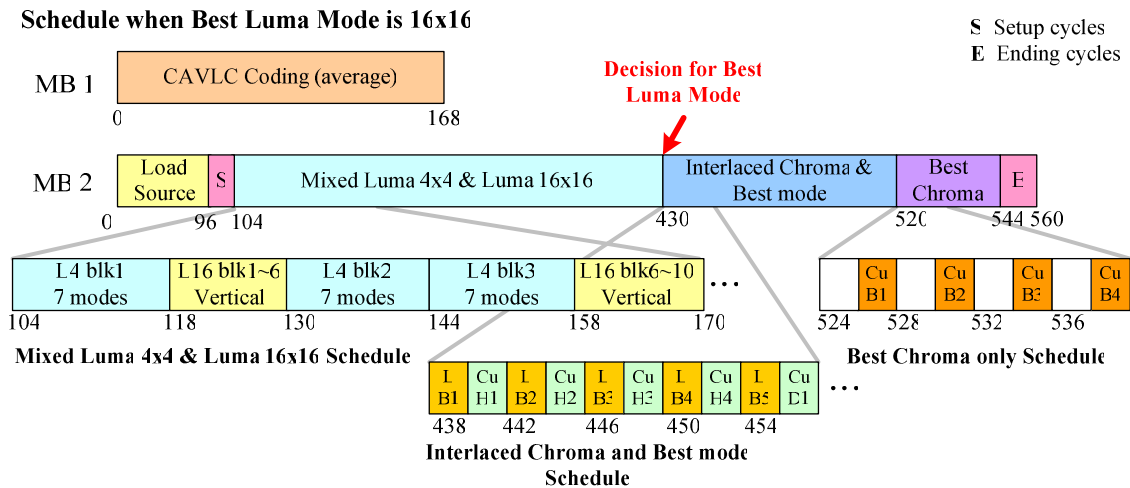


Fig. 43 Pipelined schedule for fast encoder when best luma mode is selected to 16x16

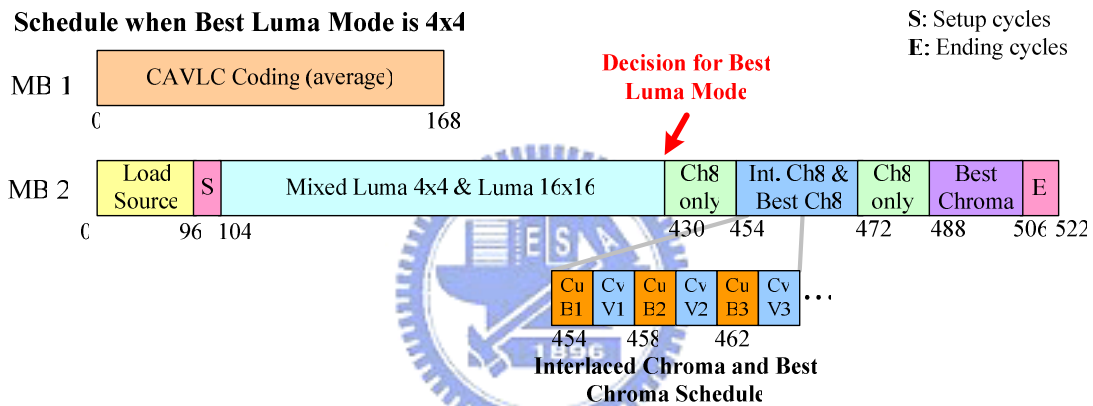


Fig. 44 Pipelined schedule for fast encoder when best luma mode is selected to 4x4

### 4.2.3. Eight-pixel Parallel Datapath

, With the eight-pixel parallelism in the prediction phase, the components in this phase have to be modified for eight inputs and eight outputs. To simplify the design and reuse previous components, our modification strategy is to use two four-pixel parallel units for eight-pixel parallelism. Fig. 45 shows the eight-pixel parallel intra prediction generator which is duplicated and optimized with removal of redundant datapath. To simplify the circuit, it only uses odd-row datapath to generate average DC values for eight outputs. The same optimization is also applied to the circuits of neighboring

reference pixel selection before the prediction engine.

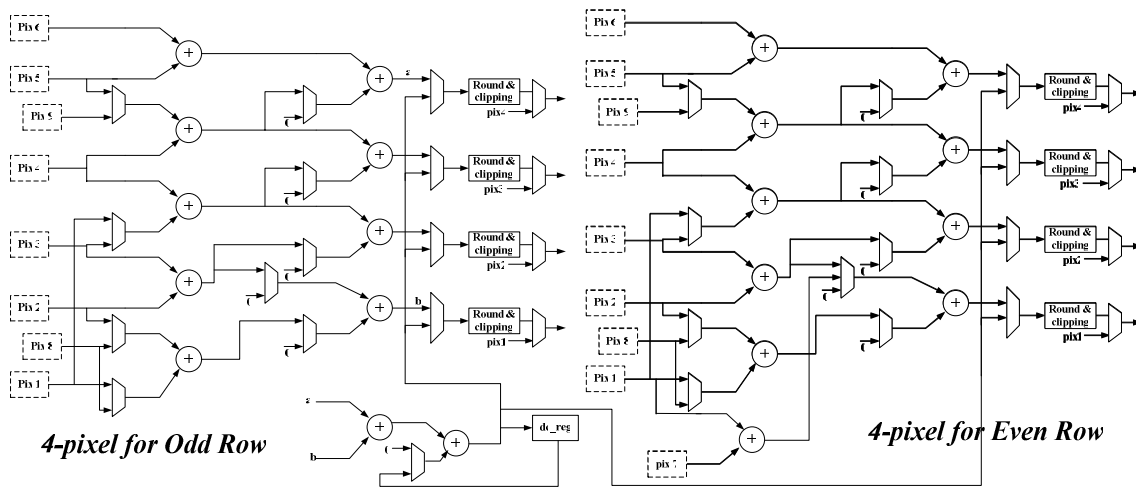


Fig. 45 Eight-pixel parallel intra prediction generator

Another primarily modified component is the 4x4 integer transform unit composed of a row-transform, a column-transform, and a 4x4 transpose array as shown in Fig. 25. First, eight inputs for two separate rows imply two separate 1-D row-transform units and two column-transform units. Then the transpose array should temporarily store the eight coefficients for two cycle latency to gather a whole block and transpose it to the right direction. Based on the previous architecture, a modified 4x4 transform unit for eight-pixel parallel architecture is shown in Fig. 46, with a 2x2x2x2 transpose array. This array delays the coefficients for two cycles and changes their propagation direction automatically. Final output data is selected by the multiplexers. The white and dark components in Fig. 46 represent different rows, and the solid and dotted arrow line means two propagation directions.

The other components in the prediction phase such as mode decision unit, cost generator, and reconstruction FIFO registers are slightly modified. Though doubling the datapath of these components may raise the area cost, some critical paths are relaxed

due to short cycle count and longer cycle delay time and thus some pipeline stages can be eliminated.

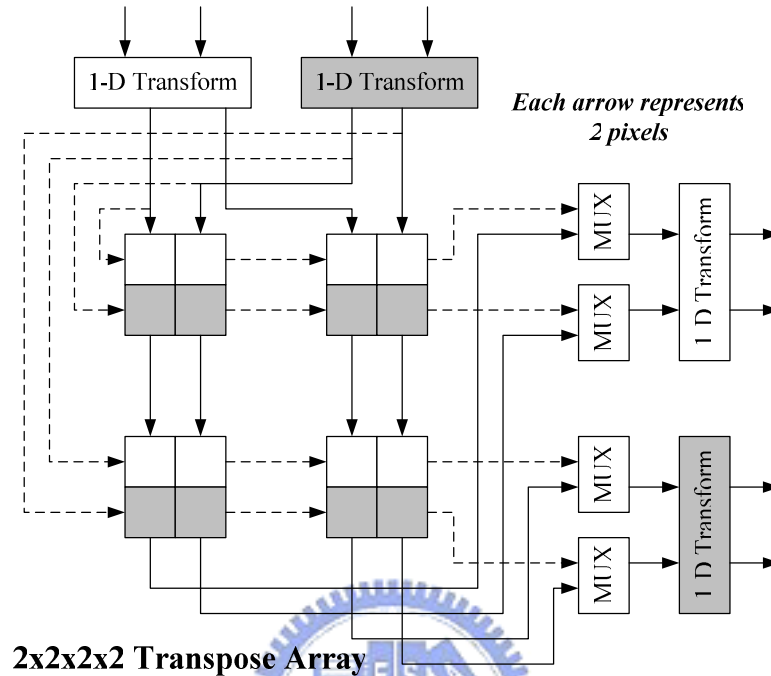


Fig. 46 Eight-input eight-output 4x4 transform unit

#### 4.2.4. Memory Organization

Fig. 47 shows the organization of two memories in this work for storage of a macroblock. Unlike the previous design, a 48-entry 64-bit single-port SRAM is used as source buffer memory to satisfy the access of eight-pixel parallel architecture instead of the 96-entry 32-bit one. The source buffer can store eight pixels per entry, as well as two rows in a block, and has 32 entries for luma and 16 for chroma components. However, the input interface of this buffer is set to four pixels, 32 bits, to reduce the bandwidth of the entire design. Thus, the cycle count to move data from external memory to the source buffer is 96, and each entry needs two cycles to be stored. The coefficient buffer is similar to previous design but decreases 25% of bandwidth to save area cost. It adopts

104-entry 48-bit ping-pong architecture to save four 12-bit quantized coefficients derived from quantization phase.

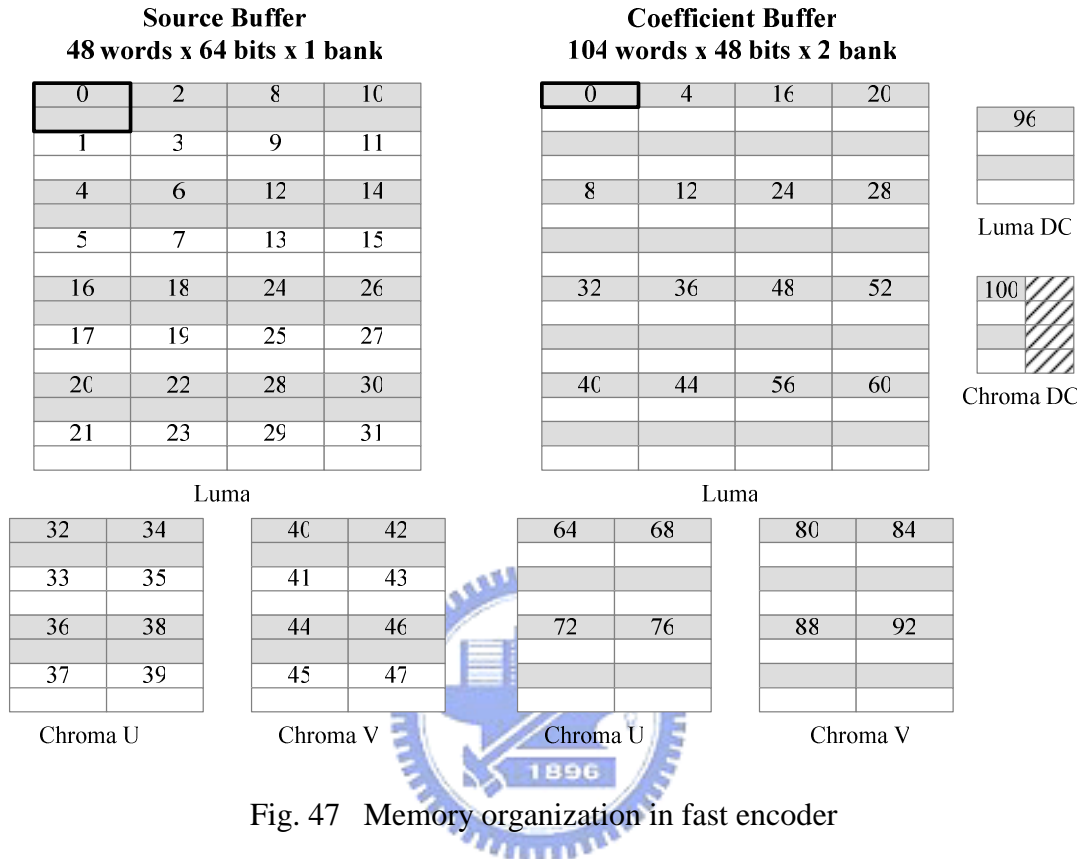


Fig. 47 Memory organization in fast encoder

#### 4.2.5. Strategies for Low Power Design

The usage of the fast prediction algorithm and variable-pixel parallel datapath not only decreases the operation cycles and frequency but saves power consumption. To achieve requirement of low power issues in portable video application, some techniques are used in both algorithm level and structure level in this design.

##### 1. Local registers for source buffer

In the schedule mentioned in Subsection 4.2.2, a block is predicted by 4x4 prediction successively for seven modes. To generate seven residual blocks, the source buffer with bandwidth of eight pixels will be read iteratively for seven times and consume

unnecessary power. To solve this problem, a local eight-pixel registers is placed between the buffer output and residual generator. The registers can hold previous output data temporarily when the buffer outputs data of the next rows, which can avoid the iterative state transient in source memory. For example, in the first mode of the 4x4 prediction, the source memory outputs two eight-pixel data. Then in the following six modes, the local register statically holds the first eight-pixel data while the memory is fixed to send last eight-pixel data. Thus, only one memory output transient is needed for seven prediction modes.

## **2. Early termination for 16x16 prediction**

There are three luma 16x16 modes inserted in the prediction schedule. However not all of them are useful for final decision. When the cost of the first-predicted 16x16 mode is obtained, the early termination for second-predicted and third-predicted modes is asserted. If the currently accumulated cost in the prediction is larger than the previous one, the following operations of this mode will be canceled and the other prediction will begin. This strategy can help to reduce redundant prediction cycles and extra power consumption.

## **3. Data guarding pipeline**

Some intra prediction modes are not allowed due to their unavailable boundary samples, such as vertical mode in the upper-most macroblock. Thus, the pipeline registers are locked for these modes and the control signals are disabled to make sure the values behind this stage are unchanged. This technique can save the unnecessary power consumption in prediction engine, transform unit, and cost generator for the invalid modes.

## 4.3. Implementation Results

### 4.3.1. Gate-count and Layout

Table 13 List of gate count for fast encoder

Component	Gate Count
Intra prediction generation	3,691
DCT/DHT with DC registers	10,008
IDCT/IDHT with DC registers	7,581
Quantization	8,832
De-quantization	2,886
Cost generation and mode decision	9,285
Reconstruction	3,840
Boundary prediction buffer	11,674
Schedule controller	1,486
CAVLC encoder	9,664
Memory controller	1,401
<b>Total</b>	<b>72,062</b>

The proposed variable-pixel parallel intra frame encoder with fast algorithm is designed by Verilog HDL and implemented using UMC 0.18 $\mu$ m 1P6M CMOS technology. When synthesizing at 62.5MHz, the total gate count is about 72K excluding the memory area. Table 13 lists the final results of gate count for each components. Though the usage of eight-pixel parallelism increases number of necessary functional units, the final gate count does not increase but decrease with the reduction of pipeline stage and relaxed critical path when compared to Table 9 . For example, the cost generator is reduced from 15.3K to 9.3K, and the quantization unit is reduced from 16.5K to 8.8K as well. Moreover, the datapath of intra prediction unit is duplicated but only has 13% of gate count increase. Most of the area is spent on boundary prediction buffer, forward transform, CAVLC encoder, and cost generator for mode decision as

shown in Table 13

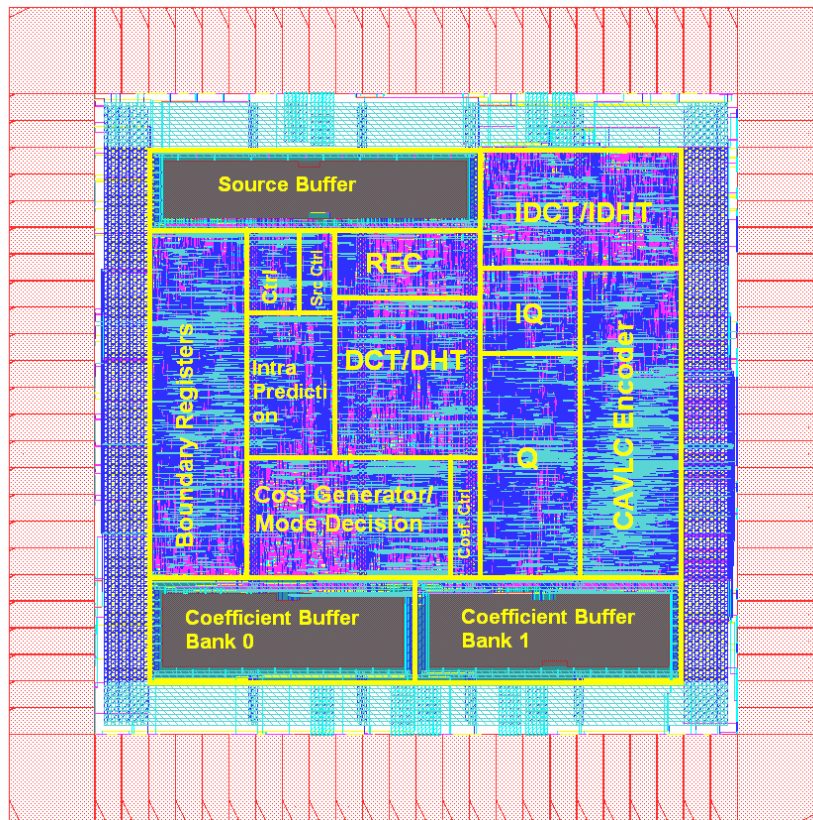


Fig. 48 Layout of the fast encoder chip

Table 14 Information for the encoder chip

<b>Technology</b>	UMC 0.18 $\mu\text{m}$ 1P6M CMOS
<b>Core Voltage</b>	1.8V
<b>I/O Voltage</b>	3.3V
<b>Core Size</b>	1.20x1.20 $\text{mm}^2$
<b>Package</b>	144 pin CQFP
<b>On-chip Memory</b>	Single-port 104 x 48-bit x 2 banks
	Single-port 48 x 64-bit x 1 bank

The chip layout is shown in Fig. 48 with a core size of 1.20x1.20  $\text{mm}^2$ , which is 13% smaller than the presented codec chip. Though the 48x64-bit SRAM is larger than the 96x32-bit one used in previous work, the total area of three SRAMs is only about 0.27  $\text{mm}^2$  instead of original 0.30  $\text{mm}^2$  and has 10% of reduction. The chip has highest



frequency at 62.5MHz and exceeds the requiring 61MHz for HD 720p encoding. Table 14 lists the chip information.

### 4.3.2. Comparison

With 30% of gate count reduction and 48% of frequency decrease compared to previous work, the proposed fast intra frame encoder can support the same size of HD 720p encoding at only 61MHz. For SD size, the required working frequency of 23MHz is also much lower than that of previous 43MHz and 54MHz in [13]. Some techniques are used to shorten the prediction cycles such as modified three-step algorithm and interlaced scheduling. The reduction of coefficient memory bandwidth helps diminish the chip area as well. Detail results of comparison are listed in Table 15

Table 15 Comparison among previous codec in Chapter 3 , [13], and this work

Design Feature	This Work	Previous Codec	[13]
Max operation freq.	62.5MHz	125MHz	55MHz
System pipeline	MB-based	MB-based	MB-based
Pixel parallelism	8 pixels/4 pixels	4 pixels	4 pixels
CMOS technology	UMC 0.18 $\mu$ m	UMC 0.18 $\mu$ m	TSMC0.25 $\mu$ m
Gate count	72K	103K	85K
Chip core size	1.20x1.20 mm <sup>2</sup>	1.28x1.28 mm <sup>2</sup>	1.86x1.86 mm <sup>2</sup>
On-chip memory usage	Single 48x64(x1)	Single 96x32(x1)	Single 96x32(x2)
	Single 104x48(x2)	Single 104x64(x2)	Single 64x32(x1)
			Dual 96x16(x4)
Max target size	HD 1280x720	HD 1280x720	HD 720x480
Freq. for HD 720p	61MHz	117MHz	N/A
Freq. for SD	23MHz	43MHz	54MHz
Freq. for CIF	6.7MHz	12.8MHz	15.8MHz
Processing cycles/MB	< 560 cycles	< 1080 cycles	< 1300 cycles
Cost Function	Enhanced DCT-based SATD	Enhanced DCT-based SATD	DCT-based SATD
Mode decision method	Modified 3-step	Full search	Full search

	fast algorithm		
Plane buffer removal	Yes	Yes	No
DC values forwarding	Yes	Yes	No
Early termination	Yes	No	No
Interlaced scheduling	Yes	No	No
Local source register	Yes	No	No
Data guarding pipeline	Yes	No	No
DCT transpose array	2x2x2x2 12-bit	4x4 16-bit	4x4 16-bit
Schedule controller	Counter-based	FSM	FSM
Bit-width for quantized coefficients	12 bits	16 bits	16 bits
Reconstruction buffer	8-4 FIFO register	FIFO register	96x32 SRAM
Coefficient register size	4x4 block	4x4 block	16x16 MB



## Chapter 5

### Conclusion

The contribution of this thesis can be divided in two parts. In Chapter 3 an MPEG-4 H.264/AVC baseline profile intra frame codec design is presented to support high definition size of 1280x720 encoding and 1920x1080 decoding at 30 frames per second. To optimize the coding process but still maintain the coding performance, this codec design adopts various techniques in both algorithm and architecture level, such as enhanced cost function, plane mode removal, and macroblock pipelining, which leads to the reduction of on-chip memories and operation cycles. In comparison with other existing design, the proposed codec has ability to save about 20% of frequency requirement for encoder and 40% for decoder with almost the same PSNR and little bitrate increase.

In Chapter 4 another H.264/AVC intra frame encoder with modified fast prediction algorithm and variable-pixel parallel architecture is proposed. Compared to previous design, this work has improvements in 48% of frequency saving, 30% of gate count decrease, 13% of chip area reduction, and 10% of memory area saving. Besides, the power consumption of this work can also be reduced by the usage of low-power techniques and less operation frequency. This design is very suitable for the products with demand of low power and portable issues, such as digital video recorder or digital still camera.

In the future work, the proposed designs can be further integrated into the baseline encoder or codec for both intra-frame and inter-frame encoding. Moreover, the CABAC

entropy coding in main profile or recently proposed intra 8x8 predictions in high profile can be added to the presented works to enhance the coding performance and image quality. We hope that this research results can promote the improvement of video application and convenience of human life as well.



## Reference

- [1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, *ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC*, in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, *JVT-G050*, Mar. 2003
- [2] Generic Coding of Moving Picture and Associated Audio Information – Part 2: Video, *ITU-T Recommendation H.262 and ISO/IEC 13818-2*, Draft International Standard, Nov. 1994
- [3] Coding of Audio-Visual Objects - Part 2: Visual, *ISO/IEC 14496-2*, International Standard:1999/Amd1:2000, Jan. 2000
- [4] A. Puri, X. Chen, A. Luthra, “Video Coding Using the H.264/MPEG-4 AVC Compression Standard,” *Signal Proc. Image Communication*, vol. 19, pp.793-849, 2004
- [5] Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5Mbits/s, *ISO/IEC 11172-2*, International Standard, Nov. 1992
- [6] Video Coding for Low Bit Rate Communication, *ITU-T Recommendation H.263 version 1*, Mar. 1996
- [7] Draft Text of Recommendation H.263 version 2 (“H.263+”) for Decision, *ITU-T Recommendation H.263 version 2*, Jan. 1998
- [8] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, “Performance comparison of video coding standards using Lagrangian coder

- control,” *Proceedings of IEEE International Conference on Image Processing* 2002, vol. 2, pp.501-504
- [9] T. Halbach, “Performance comparison: H.26L intra coding vs. JPEG2000”, *JVT-D039*, July, 2002
- [10] H.264/MPEG-4 AVC *reference software*, JM8.6
- [11] C.-C. Cheng, C.-W. Ku, and T.-S. Chang, “A 1280x720 pixels 30 frames/s H.264/MPEG-4 AVC intra encoder,” *Proc. IEEE International Symposium on Circuits and Systems*, May. 2006
- [12] C.-W. Ku, C.-C. Cheng, G.-S. Yu, M.-C. Tsai, and T.-S. Chang, “A high definition H.264/AVC intra frame codec IP for digital video and still camera applications (Accepted for publication),” *IEEE Transactions on Circuits and System for Video Technology*, to be published
- [13] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, “Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 378-401, vol. 15, no. 3, 2005
- [14] H.-M. Wang, C.-H. Tseng, and J.-F. Yang, “Improved and fast algorithm for intra 4x4 mode decision in H.264/AVC,” *Proc. IEEE International Symposium on Circuits and Systems*, May. 2005
- [15] T.-C. Wang, Y.-W. Huang, C.-Y. Tsai, B.-Y. Hsieh, and L.-G. Chen, “Dual-block-pipelined VLSI architecture of entropy coding for H.264/AVC baseline profile,” *IEEE VLSI-TSA International Symposium*, Apr. 2005

- [16] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization with 16-bit arithmetic for H.26L," *Proc. International Conference on Image Processing 2002*, vol. 2, pp.489-492, Sep. 2002
- [17] H.-Y. Lin, Y.-C. Chao, C.-H. Chen, B.-D. Liua, and J.-F. Yang, "Combined 2-D transform and quantization architecture for H.264 video coders," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1802-1805, May. 2005
- [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," *Proc. 2005 IEEE VLSI-TSA, International Symposium on VLSI Design, Automation & Test*, Apr. 2005
- [19] K.-H. Chen, J.-I. Guo, and J.-S. Wang, "High-performance direct 2-D transform coding IP design for MPEG-4 AVC/H.264," *Proc. IEEE International Symposium on Circuits and Systems*, May. 2005
- [20] T.-C. Wang, Y.-W. Huang, H.-C. Fang, and L.-G. Chen, "Parallel 4x4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 800-803, May. 2003
- [21] T.-A. Lin, S.-Z. Wang, T.-M. Liu, and C.-Y. Lee, "An H.264/AVC decoder with 4x4-block level pipeline," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1810-1813, May. 2005
- [22] F. Fu, X. Lin, and L. Xu, "Fast intra prediction algorithm in H.264/AVC," *Proc. IEEE International Conference on Signal Processing*, vol. 2, pp. 1191-1194, Aug.

2004.

- [23] B. Meng, O. C. Au, C.-W. Wong, H.-K. Lam, "Efficient intra-prediction algorithm in H.264," *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 837-840, Sep. 2003
- [24] C.-L. Yang, L.-M. Po, W.-H. Lam, "A fast H.264 intra prediction algorithm using macroblock properties," *Proc. IEEE International Conference on Image Processing*, vol. 1, pp. 461-464, Sep. 2004
- [25] F. Pan, X. Lin, S. Rahardja, K.-P. Lim, Z.-G. Li, D. Wu, and S. Wu, "Fast intra mode decision algorithm for H.264/AVC video coding," *Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 781-784, Oct. 2004
- [26] C. Kim, H.-H. Shih, C.-C. J. Kuo, "Feature-based intra-prediction mode decision for H.264," *Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 769-772, Oct. 2004
- [27] C. Chen, P.-H. Wu, H. Chen, "Transform-domain intra prediction for H.264," *Proc. IEEE International Symposium on Circuits and Systems*, vol.2, pp. 1497-1500, May. 2005
- [28] C.-C. Cheng, T.-S. Chang, "Fast three step intra prediction algorithm for 4x4 blocks in H.264," *Proc. IEEE International Symposium on Circuits and Systems*, vol.2, pp. 1509-1512, May. 2005
- [29] W. Zouch, A. Samet, M. A. Ben Ayed, F. Kossentini, N. Masmoudi, "Complexity analysis of intra prediction in H.264/AVC," *Proc. IEEE International Conference on Microelectronics*, pp. 713-717, Dec. 2004



## 作者簡歷

姓名：古君偉

籍貫：台北市

學歷：台北市立建國高級中學 (1997年9月~2000年6月)

國立交通大學電子工程學系 學士 (2000年9月~2004年6月)

國立交通大學電子研究所 碩士 (2004年9月~2006年6月)

得獎經歷：

◆ 九十三學年度 大專院校積體電路設計競賽

研究所/大學部 標準單元式設計組 (Cell-based Design) 佳作

◆ 九十四學年度 大專院校積體電路設計競賽

研究所/大學部 標準單元式設計組 (Cell-based Design) 特優



論文著作：

Chao-Chung Cheng, **Chun-Wei Ku**, and Tian-Sheuan Chang, "A 1280x720 pixels 30 frames/s H.264/MPEG-4 AVC intra encoder," *Proc. IEEE International Symposium on Circuits and Systems*, May. 2006

**Chun-Wei Ku**, Chao-Chung Cheng, Guo-Shiuan Yu, Min-Chi Tsai, and Tian-Sheuan Chang, "A high definition H.264/AVC intra frame codec IP for digital video and still camera applications (Accepted for publication)," *IEEE Transactions on Circuits and System for Video Technology*, to be published