

MPEG-4 材質編碼器之架構設計與實現

學生：洪堯俊

指導教授：吳炳飛 教授

國立交通大學電機與控制工程學系（研究所）碩士班

摘 要

在本論文中，我們提出了一個 MPEG-4 材質編碼(texture coding)架構設計，適合用於目前的多媒體視訊應用。材質編碼是 MPEG-4 視訊編碼中重要的環節，用於去除空間與頻率領域的冗餘資料並進行壓縮編碼，可以有效地降低儲存視訊內容所需的空間，在較低頻寬的網路環境下，提供良好畫質的視訊傳輸。材質編碼的部分包括了離散餘弦轉換(Discrete Cosine Transform)、量化(Quantization)、反量化(Inverse Quantization)、反轉離散餘弦轉換(Inverse Discrete Cosine Transform)以及交流直流預測(AC/DC prediction)。首先，我們採用 DCT、IDCT 交替的排程方式來處理影像中單一巨集區塊(Macroblock)，利用運算處理單元共用的方式，有效地減少硬體面積以及節省運算處理所需的時間，再者，我們使用行列分解的技術來降低二維 DCT/ IDCT 的運算複雜度，能夠利用簡便的訊號控制同一硬體架構進行 DCT 或是 IDCT 運算，所提出的演算法能有效地減少乘法運算元並且符合 IEEE 所制定的 IDCT 精確度要求。此外，在整個材質編碼器前端，亦設計一個 ping-pong 緩衝區，使得材質編碼器的運作能夠管線(pipeline)進行，安全且正確地讀取資料進行編碼，達到較佳的編碼效能。我們所提出的材質編碼架構適用於即時視訊壓縮，可以支援 MPEG-4 Simple Profile Level 3 標準的位元流編碼。

所提出的材質編碼架構在影像大小為 CIF 的格式下，處理單一巨集區塊的時間為 1137 個時間週期，透過 UMC 0.18 製程技術合成，最大操作頻率為 43MHz，邏輯閘總數為 54,405 個，使用 16,840 位元的內部記憶體，此外，能夠輕易的整合進 MPEG-4 編

碼器中，並且適用於行動通訊應用。



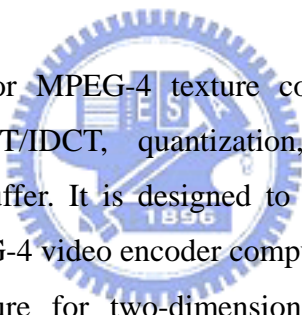
Architecture Design and Implementation of MPEG-4 Texture Coding

Student : Yao-Chun Hung

Advisor : Prof. Bing-Fei Wu

Department of Electrical and Control Engineering
National Chiao Tung University

ABSTRACT



An efficient architecture for MPEG-4 texture coding is proposed in this thesis. The architecture consists 2D-DCT/IDCT, quantization, AC/DC prediction block, inverse quantization and ping-pong buffer. It is designed to handle a macroblock data within 1137 cycles and is suitable for MPEG-4 video encoder computing CIF (352x288) image formats. The cost-effective VLSI architecture for two-dimensional 2D DCT/ IDCT is based on the row-column decomposition technique. The 2D DCT/IDCT has a regular structure which will be interconnect and control simply, and the implementation of the inverse transform can use the same hardware efficiently. In addition, the proposed 2D IDCT algorithm used only four parallel multipliers and conforms to the accuracy specification of IEEE standard 1180 -1990. Furthermore, an efficient block engine with the interleaving DCT/IDCT scheduler is achieved for scheduling DCT, quantization, inverse quantization, IDCT, and AC/DC prediction in order to reduce the hardware cost and processing time. Besides, an additional ping-pong buffer is carried out to generate two independent addresses for reading and writing at the same time. Therefore, all the data in the buffer can be read safely and correctly. A typical MPEG-4 Simple Profile Level 3 sequence can be encoded in real-time with the proposed texture coding module.

The proposed design has been synthesized by using 0.18-um CMOS technology. The simulation results indicate that MPEG-4 texture coding can run at a maximum frequency of 43 MHz and it contains 54,405 gates and 16,840 bits memory. In addition, our texture coding

engine can be integrate into the entire MPEG-4 encoder easily, and it suitable for mobile communication applications.



誌 謝

本論文得以完成，首先要感謝我的指導老師吳炳飛教授在我研究所生涯的悉心教誨，在課業研究或是為人處世各方面，老師都給予我最佳的研究環境以及學習典範，讓我在資源充沛的環境中可以充分地研究與學習，而老師對研究嚴謹與認真的態度，更成為我學習的目標。另外，承蒙蔡淳仁教授、蘇崇彥教授以及莊俊雄博士撥冗參加學生口試，並給予寶貴的建議，讓本論文更加充實完整，特此致謝。

在實驗室的生涯中，感謝忠哥(瞿忠正學長)、阿誠(陳昭榮學長)、阿霖(陳彥霖學長)、重甫、全財以及信元，在你們的帶領與教導之下，讓我不論在研究或處世方面，都有相當的成長。感謝和我一起奮鬥的東龍、則全、PPJ(林裕傑學弟)，能與你們一起合作與研究，讓我感到十分地榮幸與快樂。感謝晏阡與培恭，在硬體設計上的問題，都能夠不吝地給予協助。另外，感謝小熊(黃嘉雄學弟)、宗堯、元馨、晉源、秉宗，感謝你們在生活上給予的幫助。

最後，我要致上最深的感謝給我最親愛的父母親以及家人，因為有你們的鼓勵與支持，辛勤地付出讓我生活無虞，使我能夠順利的完成學業，雖然因忙於課業而無法時常陪伴在你們身邊，但對你們的愛與關懷仍舊不變。

謹將此文獻給我最親愛的家人以及所有關心與愛護我的朋友們

洪堯俊 94年7月

Table of contents

摘要.....	I
ABSTRACT	III
誌謝.....	V
TABLE OF CONTENTS	VI
LISTS OF FIGURE.....	VIII
LISTS OF TABLE	X
AWARD	XI
CHAPTER 1 INTRODUCTION.....	1
1.1 BACKGROUND	1
1.2 MPEG-4 STANDARD OVERVIEW.....	2
1.3 THESIS ORGANIZATION.....	5
CHAPTER 2 OVERVIEW OF MPEG-4 VIDEO CODING.....	6
2.1 INTRODUCTION	6
2.2 MOTION ESTIMATION.....	8
2.3 TEXTURE CODING.....	9
2.3.1 DCT and inverse DCT.....	10
2.3.2 Quantization and inverse quantization.....	11
2.3.3 AC/DC Prediction.....	14
2.3.4 Scan	15
2.3.5 Variable Length Coding.....	16
CHAPTER 3 ARCHITECTURE DESIGN OF MPEG-4 VIDEO TEXTURE CODING.....	17
3.1 TEXTURE CODING UNITS OVERVIEW	17
3.2 PING-PONG BUFFER UNIT	19
3.3 DCT/IDCT ARCHITECTURE.....	20
3.4 QUANTIZATION AND INVERSE QUANTIZATION DESIGN.....	29
3.5 AC/DC PREDICTION DESIGN	33
3.6 FINITE STATE MACHINE OF TEXTURE CODING.....	39
CHAPTER 4 ASIC IMPLEMENTATION.....	45
4.1 DESIGN FLOW	45
4.2 FUNCTIONAL VERIFICATION.....	46

4.3 IMPLEMENTATION RESULT	50
CHAPTER 5 CONCLUSION	57
REFERENCE	59
APPENDIX	61
A-1 . PIN DEFINITIONS FOR MPEG-4 TEXTURE CODING	61
BIOGRAPHY	65



Lists of Figure

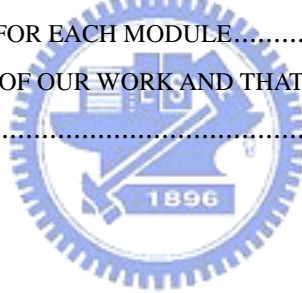
FIGURE 2-1 BASIC CODING FLOW OF MPEG-4.....	7
FIGURE 2-2 BLOCK MATCHING ALGORITHM	8
FIGURE 2-3 BLOCK DIAGRAM OF MPEG-4 TEXTURE CODING.....	10
FIGURE 2-4 DEFAULT WEIGHTING MATRICES FOR MPEG QUANTIZATION.	13
FIGURE 2-5 AC/DC PREDICTION PROCESS FOR INTRA CODED MBS.	14
FIGURE 2-6 ALTERNATE MPEG-4 SCANNING MODE	16
FIGURE 3-1 OVERALL ARCHITECTURE OF MPEG-4 TEXTURE CODING	17
FIGURE 3-2 ARCHITECTURE OF MPEG-4 TEXTURE CODING DATA PATH	18
FIGURE 3-3 ARCHITECTURE OF PING-PONG BUFFER	20
FIGURE 3-4 2-D DCT/IDCT ARCHITECTURE OF ROW-COLUMN DECOMPOSITION.....	22
FIGURE 3-5 ARCHITECTURE OF 1D DCT/IDCT UNIT.....	23
FIGURE 3-6 ARCHITECTURE OF MULTIPLIER-ADDER UNIT	23
FIGURE 3-7 READ/WRITE ACTION OF DCT/IDCT TRANSPOSE MEMORY	27
FIGURE 3-8 BLOCK DIAGRAM OF A MULTIPLIER-ADDER BASED 2-D DCT/IDCT.....	28
FIGURE 3-9 ROUNDED MULTIPLICATION DOT DIAGRAM	29
FIGURE 3-10 FLOWCHART OF THE QUANTIZATION ARCHITECTURE.....	31
FIGURE 3-11 CODED BLOCK PATTERN IN INTRA CODING MODE	32
FIGURE 3-12 FLOWCHART OF INVERSE QUANTIZER.....	33
FIGURE 3-13 Y COMPONENT STORAGE FOR AC/DC PREDICTION.....	34
FIGURE 3-14 STRUCTURE OF PREDICTION MEMORY	34
FIGURE 3-15 TIMING DIAGRAM OF INTERLEAVED DCT/IDCT SCHEDULING	35
FIGURE 3-16 ARCHITECTURE OF AC/DC PREDICTION.....	36
FIGURE 3-17 AC/DC PREDICTION FINITE STATE MACHINE.....	38
FIGURE 3-18 AMBA READ/WRITE FINITE STATE MACHINE.....	40
FIGURE 3-19 FINITE STATE MACHINE OF TEXTURE CODING	41
FIGURE 3-20 INTRA ENCODING PROCEDURE.....	42
FIGURE 3-21 INTER ENCODING PROCEDURE.....	43
FIGURE 3-22 BLOCK-LEVEL OPERATION PROCEDURE.....	44
FIGURE 4-1 MODULE DESIGN FLOW	46
FIGURE 4-2 FLOW CHART OF FUNCTIONAL VERIFICATION	47
FIGURE 4-3 OVERVIEW OF DFT COMPILER FLOW	48
FIGURE 4-4 MEMORY BUILT-IN SELF-TEST ARCHITECTURE	49
FIGURE 4-5 SETUP FOR MEASURING THE ACCURACY OF A PROPOSED 8x8 IDCT	50
FIGURE 4-6 TIMING OF THE PROPOSED ARCHITECTURE	51
FIGURE 4-7 GATE COUNT OF EACH MODULE.....	53

FIGURE 4-8 SUBJECT VIEW OF RECONSTRUCTED FRAME FOR AKIYO SEQUENCES AT THE 46TH FRAME AND	55
FIGURE 4-9 SUBJECT VIEW OF RECONSTRUCTED FRAME FOR FOREMAN SEQUENCES AT THE 27TH FRAME AND 109 TH FRAME.....	55
FIGURE 4-10 SUBJECT VIEW OF RECONSTRUCTED FRAME FOR TABLE SEQUENCES AT THE 122ND FRAME AND 183 RD FRAME	56
FIGURE 4-11 SUBJECT VIEW OF RECONSTRUCTED FRAME FOR NEWS SEQUENCES AT THE 21 ST FRAME AND 146 TH FRAME.....	56
FIGURE A-1 MPEG-4 TEXTURE CODING IP.....	61



Lists of Table

TABLE 1-1 LISTS OF VIDEO/IMAGE CODING STANDARDS	2
TABLE 1-2 MPEG-4 VISUAL PROFILES AND CORRESPONDING TOOLS	4
TABLE 2-1 AC/DC PREDICTION PSEUDO CODE.....	15
TABLE 3-1 COMPLEXITY OF THE PROPOSED 1-D DCT/IDCT ALGORITHM	26
TABLE 3-2 COMPLEXITY OF DIFFERENT 1D DCT/IDCT DESIGN.....	26
TABLE 3-3 THE OPTIMIZED WORDLENGTH FOR THE OUR 2-D DCT/IDCT ARCHITECTURE	28
TABLE 3-4 CORE CHARACTERISTICS OF THE DCT/IDCT ARCHITECTURE.....	29
TABLE 3-5 NON LINEAR SCALER FOR DC COEFFICIENTS OF DCT BLOCKS,	30
TABLE 3-6 H.263 QUANTIZER TO GUARANTEE ALL COEFFICIENTS EQUAL ZERO.	31
TABLE 3-7 STORING LEFT TOP VALUE AND READING FOR EACH BLOCK INDEX.....	35
TABLE 4-1 ACCURACY TEST RESULT OF THE IDCT	50
TABLE 4-2 COMPARISON WITH PREVIOUS WORK	52
TABLE 4-3 MEMORY REQUIRED FOR EACH MODULE.....	52
TABLE 4-4 THE AVERAGE PSNR OF OUR WORK AND THAT OF THE XVID VERSION.....	54
TABLE A-1 PIN DEFINITION	62



Award

I. 本論文曾經參與 研華文教基金會 第六屆 TIC100 科技創新事業競賽冬令營 亞軍



II. 本論文曾入圍 2005 年教育部矽智產(SIP)設計競賽決賽

Chapter 1 Introduction

1.1 Background

Video source coding is developed for over 20 years, and many video compression techniques have provided to improve the video coding efficiency. The primary purpose in the design of video coding system is to reduce the transmission rate and promote the quality. Because the storage requirement of original video sequences is very large, it is required to reduce data by compressing. Two approaches to achieve this propose are removal of statistic redundancy and psychophysical redundancy of video sequence. In the statistic redundancy, the video sequence is usually high-correlation in the spatial or temporal domain. The dependency in the spatial and temporal domain can be used to predict and get the statistic redundancy of the video sequences. The statistic redundancy such as the motion estimation and AC/DC prediction is widely-used in many video standards, for instance, MPEG-4[1] and H.263 [2]. Since the video system is based on the human observation in the psychophysical redundancy, the human vision is not sensitive to high frequency. We can employ the perceptive limitation of the human version to reduce the transmission requirements. The lossy compression techniques like quantization are provided to achieve the psychophysical redundancy without affecting perception, or with little reduction which could be disregarded.

To drive the fast development of multimedia industry, the standards of digital video coding are specified. Several video standards have been made during the past few years. Table 1-1 lists the roadmap of the video/image coding standards. A novel video compression technique, MPEG-4, is introduced in the next section.

Table 1-1 lists of video/image coding standards

standards	International standard	Main features
ISO JEPG	1991	Continuous-tone still image, DCT based
ITU-T H.261	1990	Low bit-rate video conferencing, px64kbps
ISO/IEC MPEG-1	1992	Video CD (storage), 1.5Mbps
ISO/IEC MPEG-2	1994	Digital TV(Broadcasting), 2~15Mbps
ITU-T H.263	1995	Very low bit-rate coding, < 64 kbps
ITU-T H.263+	1998	Add many advanced coding options to H.263
ISO/IEC MPEG-4	1999	Multimedia communication, content-based coding
ISO/IEC JPEG-2000	2000	Still image coding, DWT coded
ISO/IEC MPEG-4(v2)	2000	Add more tools and profiles to MPEG-4
ITU-T H.263++	2000	Add more advanced coding options to H.263++
ITU-T H.26L	2001	Functionality different, much more efficient
ISO/IEC MPEG-4(v3)	2001	Extend more tools/profiles to MPEG-4

1.2 MPEG-4 standard overview

MPEG-4 is well-known as the functionality-rich and high flexible multimedia standard. It is an ISO/IEC standard developed by MPEG (Moving Picture Expert Group), the committee that also developed the Emmy winning standards known as MPEG-1[3] and MPEG-2[4]. MPEG-4 wants to address a wide range of applications, and many of them are completely new. MPEG-4 does not target a major and exclusive killer application but opens many new frontiers. New and richer applications are developed, for instance, enhanced broadcasting, remote surveillance, personal communications, games, mobile multimedia, and virtual

environments [5]. The MPEG-4 has developed eight new or improved functionalities to support these applications, special in the three worlds-TV/film/entertainment, computing, and telecommunications. These functionalities can be classified into three categories [6], [7].

Chief among them on the following:

- 1). **Compression efficiency** : This class includes functionalities for coding of multiple concurrent data streams and improved coding efficiency. These functionalities are needed for all applications relying on efficient transmission or the storage of video data. One example of such applications is the video transmission over IP.
- 2). **Content-based interactivity** : These are functionalities to allow for content- based access and manipulation of data, editing bit streams, coding hybrid (natural and synthetic) data, and improved temporal random access. The functionalities will target the applications such as electronic shopping, digital library, and movie product.
- 3). **Universal access** : Such functionalities consist of robustness in error-prone environments and content-based scalability. These functionalities allow MPEG-4 encoded data to be accessible over a wide range of media, with various qualities in terms of temporal and spatial resolutions for specific objects. These different resolutions can be decoded by a range of decoders with different complexities. Applications benefiting from them are mobile communications, database browsing, and access at different content levels, scales, qualities, and resolutions.

In the MPEG-4 standards, profiles determine the tool set. For a given profile, a level defines quantitative bounds on technical parameters in order to bound implementation complexity and cost. Profiles will be convergence points for industry standards built on MPEG-4. Table 1-2

shows the definitions of visual profiles in the MPEG-4 standard. Each profile contains many coding tools to support the specific applications. The simple profile provides error robust coding of rectangular video objects for low bit rate applications such as mobile communications. The simple scalable profile adds capabilities for temporal and spatial scalability. It can be used in the networks with variable bit rates. The main profile supports interlaced coding and semi-transparent arbitrarily shaped objects, and its applications include entertainment and broadcast. The core profile provides better visual quality by using B-VOP, and supports to code the arbitrarily shape video objects. The major applications of the core profile are internet multimedia applications.

Table 1-2 MPEG-4 visual profiles and corresponding tools

Visual Tools		Simple Profile	Simple Scalable Profile	Core Profile	Main Profile	Advanced Coding Efficiency Profile
Basic	I-VOP	✓	✓	✓	✓	✓
	P_VOP	✓	✓	✓	✓	✓
	AC/DC Prediction	✓	✓	✓	✓	✓
	4MV, Unrestricted MV	✓	✓	✓	✓	✓
Error Resilience	Slice Resynchronization	✓	✓	✓	✓	✓
	Data Partitioning	✓	✓	✓	✓	✓
	Reversible VLC	✓	✓	✓	✓	✓
	Short Header	✓	✓	✓	✓	✓
B-VOP			✓	✓	✓	✓
Method1/Method2 Quantization				✓	✓	✓
P-VOP Based Temporal Scalability	Rectangular			✓	✓	✓
	Arbitrary Shape			✓	✓	✓
Binary Shape				✓	✓	✓
Grey Shape					✓	✓

Interlace				✓	✓
Sprite				✓	
Temporal Scalability (Rectangular)		✓			
Spatial Scalability (Rectangular)		✓			
Global Motion Compensation					✓
Quarter-pel Motion Compensation					✓
SA-DCT					✓

1.3 Thesis organization

The organization of the thesis is depicted as follows. This chapter is a brief introduction to the video compression and MPEG-4 Standard. In chapter 2, MPEG-4 video coding algorithms including the motion compensation and the texture coding are represented. In chapter 3, the architecture of MPEG-4 texture coding units consisting of quantizer, inverse quantizer, DCT/IDCT, AC/DC prediction, and ping-pong buffer are introduced. In chapter 4, functional verification and ASIC implementation results are described. Finally, a conclusion is given in chapter 5.

Chapter 2

Overview of MPEG-4 Video Coding

2.1 Introduction

In this chapter, technical overview of MPEG-4 video coding will be introduced. MPEG-4 video coding supports two main coding modes : the coding of rectangular video objects and the coding of arbitrarily shaped video objects. The coding of rectangular video is described in this section. In principle, a rectangular video object is alike a frame that is used in well-known video coding standards such as MPEG-1 and MPEG-2, H.261 and H.263. Each frame of a video sequence is considered as a VOP (Video Object Plane), which is the instance of video objects at a given time. The coding of a frame is block-based—that is, each frame is divided into the MB (macroblocks) of 16x16 pixels in size. There are two major coding modes in the video encoding process, inter coding mode and intra coding mode. In inter coding mode, the ME (Motion Estimation) finds the most similar MB between the current VOP and the previous reconstructed VOPs. The most similar MB is regarded as the predicted MB. The different of MB luminance and chrominance values obtained from current VOP with respect to prediction values from previous or future VOPs is coded. The transform coding adopted by MPEG-4 video coding is the DCT (Discrete Cosine Transform). In intra coding mode, the luminance and chrominance values of MB are coded independently of previous or future VOPs and the DCT is applied to it without prediction. DCT transforms a signal or image in spatial domain into coefficients in frequent domain. Then these coefficients will be quantized (Q) by a quantized number. Two processing paths would be followed after the quantization. One processing path includes inverse quantization (IQ) and inverse discrete cosine transform

(IDCT). To avoid mismatch between the encoding process and the decoding process. This path is a decoding path which has the same process as the decode side and can get the reconstructed frame. The other one is the coding process. The AC/DC prediction will be used to get the predicted values from its neighbor MBs in intra coding mode. Due to the distribution of the coefficients in the frequency domain, scan block provides three scan methods to reorder the coefficients. They are the alternate-horizontal scan, the alternate-vertical scan, and the zigzag scan. The reordered coefficients will be input into the variable length coding (VLC), including running length coding and Huffman coding. The overall coding flow is shown in Fig. 2-1. In the intra coding mode, the motion compensation (MC) is not considered. The values of video sources will be directly input into DCT and the reconstructed values of the Frame Memory will only be from IDCT. In the inter coding mode, motion compensation errors, which are obtained by subtracting the predicted MB from the source MB, are used for inputting into DCT. The reconstructed values are the summation of the predicted values and the motion compensation errors.

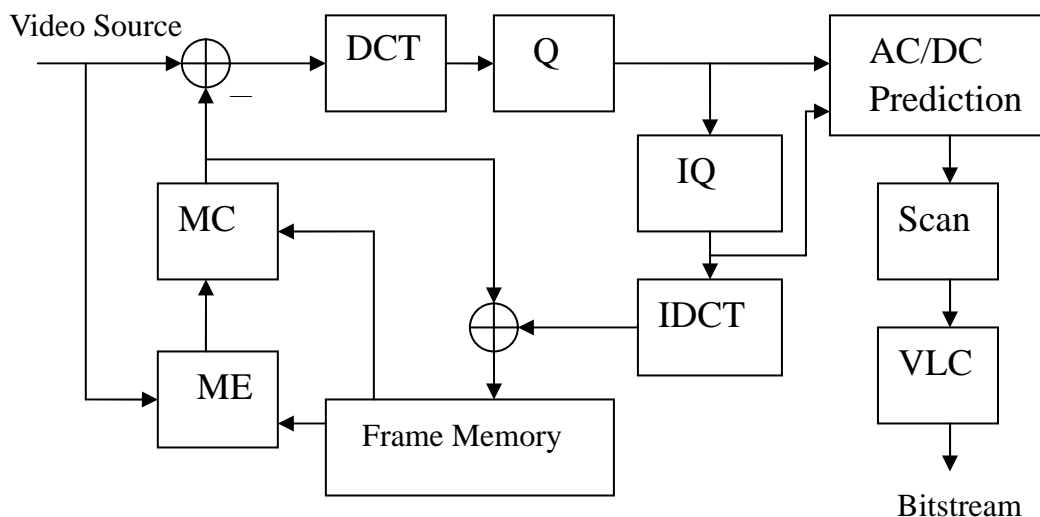


Figure 2-1 Basic coding flow of MPEG-4

2.2 Motion Estimation

Motion estimation is the key technique of many video compression schemes to remove temporal redundancy and improve coding efficiency. To compress the video, the temporal redundancy between adjacent frames could be exploited. Therefore, a reference frame is selected, and subsequent frame is predicted from the reference frame using the motion estimation. Block matching is the most common method of motion estimation.

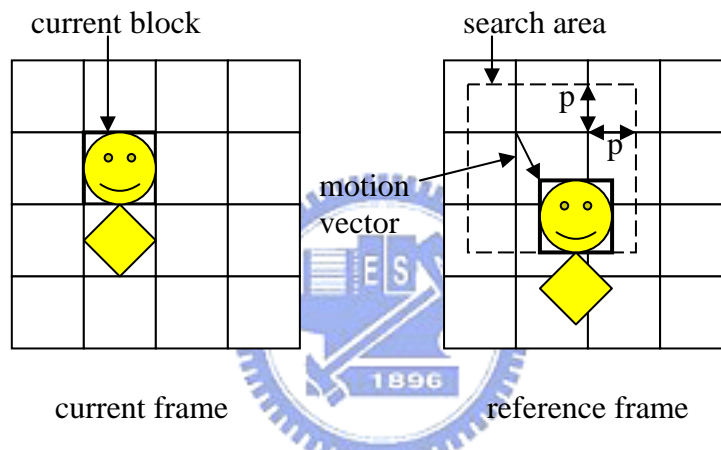


Figure 2-2 Block matching algorithm

The previous decoded frame is regarded as the reference frame. Each macroblock in the source frame is compared with shifted regions of the same size from the reference frame, and we have to search the most similar macroblock in the reference frame. Due to the computational complexity, we could define different areas in the reference frame as search range. As shown in Fig. 2-2, $[-p, p-1]$ is the search range. The displacement which results in the minimum mismatch error is selected as the best MV (motion vector) for this macroblock. SAD (sum of absolute difference) is the most widely-used method to judge which block is the best match. The equation of SAD is shown as (2-1).

$$SAD(u, v) = \sum_{i=1}^N \sum_{j=1}^N |c(i, j) - ref(i, j, u, v)|, \text{ for } -p \leq u, v \leq p-1 \quad (2-1)$$

Where i is the horizontal pixel index and j is the vertical pixel index in the current block. $Ref(i, j, u, v)$ is the pixel in the reference block. Motion vectors are (u, v) if $SAD(u, v)$ is the minimum SAD in search area. During the motion compensation, the MB in the reference frame that is referenced to by the motion vector is copied into the reconstructed frame. The mismatch error between the current block and the reference block will be delivered to both the decode side and the reconstructed frame to compensate the error.

2.3 Texture Coding

This section describes the texture coding technique in the MPEG-4. The texture coding in the MPEG-4 video has two coding modes: one is the coding of luminance and chrominance values in the intra mode, and the other is the coding of prediction error values after the motion-compensated prediction in the inter mode. The coding process is shown in Fig. 2-3. The $f[y][x]$ denotes the video sources which are either the luminance and chrominance values or the prediction error values of an 8x8 pixels block. An 8x8 DCT transforms $f[y][x]$ in the 2-D spatial domain into $F[v][u]$ in the 2-D frequency domain. Then the transform coefficients $F[v][u]$ are quantized to $QF[v][u]$. The AC/DC prediction performs prediction of some of the transform coefficients only in intra coding mode. Finally, the scan block will convert the two-dimensional matrix of the coefficients and the prediction differences $PQF[v][u]$ into a one-dimensional vector $QFS[v][u]$. Then using a variable length coding to encoding this vector. The more detail procedure in each sub-block will be described in the following sections.

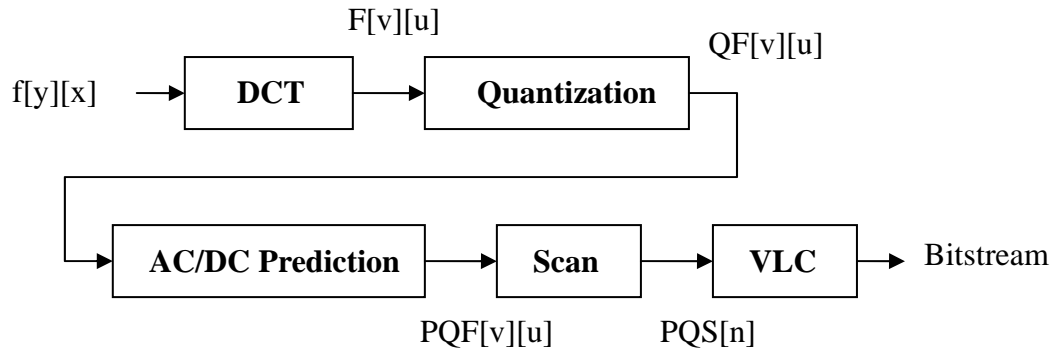


Figure 2-3 Block diagram of MPEG-4 texture coding

2.3.1 DCT and inverse DCT

The DCT (discrete cosine transform) is one of the most frequently used transformations for image compression. It transforms a signal from the spatial domain into the frequency domain. For most images, the energy almost lies at the low frequencies and the low frequencies will appear in the upper-left corner after the DCT. The values in the lower-right corner represent the high frequencies. For human vision, the values of high frequency are less sensitive, and are often – small enough to be neglected with little visible distortion. Now we describe the mathematical basis of the DCT and show how it is applied to encode an image.

◆ *Forward DCT Transform*

The block size of 8x8 for performing the 2-D DCT is small enough for the transform to be quickly computed but big enough for significant compression. For an 8x8 block of pixel values $f(x, y)$, the 2-D DCT is defined as (2-2).

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$\text{where } \alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for other } k \end{cases}, \quad 0 \leq k \leq N-1 \quad (2-2)$$

◆ **Inverse DCT Transform**

In order to have the same processing as the decoding side, an inverse transform is needed in the encoding procedure. The two-dimensional inverse discrete cosine transform (2-D IDCT) for an 8x8 block is described by

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u, v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$\text{where } \alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for other } k \end{cases}, \quad 0 \leq k \leq N-1 \quad (2-3)$$



2.3.2 Quantization and inverse quantization

DCT and IDCT are the transform pairs, and they do not yield any compression by themselves. In order to reduce the required bit rate for texture coding, subsequent quantization of the transform coefficients can compress the data.

The quantization can be viewed as division followed by integer truncation. The coefficients in higher frequencies after the DCT are usually small and quantized to zero. That will reduce redundancy and will improve the compression efficiency. The quantization is the lossy compression. That is, it will result in some distortion between its input data and output data after dequantized. The MPEG-4 supports two possible quantization procedures to quantize the DCT coefficients. The first is derived from the MPEG-2 video standard, and the second one is used in the H.263 defined by the International Telecommunication Union –

Telecommunication Standardization Sector which quantization method selection is decided at the encoder side.

A specific parameter called `quantizer_scale` decides the quantization step size. It can take the values from 1 to 31 and is coded once per VOP. In the following, both the quantization formula and the inverse quantization formula are described.

◆ ***Intra DC Coefficient Quantization***

The nonlinear quantization method is provided for the DC coefficients of intra coded macroblocks. The value of `dc_scaler` depends on the value of `quantizer_sacle` and the operator `//` denotes an integer division with rounding to the nearest integer.

$$\text{Quantization : } QF[0][0] = F[0][0] // dc_scaler \quad (2-4)$$

$$\text{Inverse quantization : } F[0][0] = QF[0][0] \bullet dc_scaler \quad (2-5)$$



◆ ***MPEG Quantization***

This quantization method is derived from the MPEG-2 video standard. It allows the users to adapt the quantization step size individually for each transform coefficient by means of weighting matrices. Since the human eye is more sensitive to low spatial frequency and less sensitive to high spatial frequencies, the transform coefficients in high spatial frequency could be quantized more coarsely than those in low spatial frequencies. The default quantization matrices for intra and inter coded macroblocks are shown in Fig. 2-4. Then we can use (2-6) and (2-7) to compute quantization coefficients and dequantized coefficients, respectively.

8	17	18	19	21	23	25	27
17	18	19	21	23	25	27	28
20	21	22	23	24	26	28	30
21	22	23	24	26	28	30	32
22	23	24	26	28	30	31	35
23	24	26	28	30	32	35	38
25	26	28	30	32	35	38	41
27	28	30	32	35	38	41	45

Default weighting matrix for
intra coded MBs

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	21
23	24	25	27	28	30	31	33

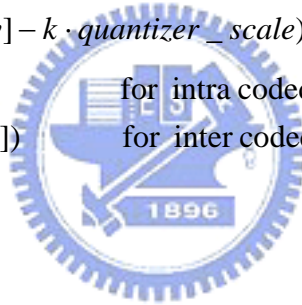
Default weighting matrix for
inter coded MBs

Figure 2-4 Default weighting matrices for MPEG quantization.

Quantization :

$$Q[u][v] = (F[u][v] \cdot 16 // W[u][v] - k \cdot \text{quantizer_scale}) / (2 \cdot \text{quantizer_scale})$$

$$\text{where } k = \begin{cases} 0 & \text{for intra coded blocks} \\ \text{sign}(QF[v][u]) & \text{for inter coded blocks} \end{cases} \quad (2-6)$$



Inverse quantization :

$$F''[u][v] = \begin{cases} 0, & \text{if } QF[v][u] = 0 \\ ((2 \times QF[v][u] + k) \times W[w][v][u] \times \text{quantizer_scale}) / 16, & \text{if } QF[v][u] \neq 0 \end{cases}$$

$$\text{where } k = \begin{cases} 0 & \text{for intra coded blocks} \\ \text{sign}(QF[v][u]) & \text{for inter coded blocks} \end{cases} \quad (2-7)$$

◆ H.263 Quantization

This quantization method is derived from the coefficient quantization used in H.263. It doesn't apply the weighting matrix technique and only quantize by a constant value. Equations of the quantization and inverse quantization are shown as follows.

Quantization :

$$|QF[u][v]| = \begin{cases} |F[u][v]| / (2 \cdot \text{quantiser_scale}) & \text{for intra coded blocks} \\ (|F[u][v]| - \text{quantiser_scale}) / (2 \cdot \text{quantiser_scale}) & \text{for inter coded blocks} \end{cases} \quad (2-8)$$

Inverse quantization :

$$|F'''[v][u]| = \begin{cases} 0, & \text{if } QF[v][u] = 0 \\ (2 \times |QF[v][u] + 1|) \times \text{quantizer_scale}, & \text{if } QF[v][u] \neq 0, \text{ quantizer_scale is odd} \\ (2 \times |QF[v][u] + 1|) \times \text{quantizer_scale}, & \text{if } QF[v][u] \neq 0, \text{ quantizer_scale is even} \end{cases} \quad (2-9)$$

2.3.3 AC/DC Prediction

There exist statistical dependencies for some of the AC and DC coefficients of the neighboring blocks. We can predict the values of one block from the corresponding values of one of the neighboring blocks. The AC/DC prediction is applied in only the case of intra coded MBs. The AC/DC prediction process is shown in Fig. 2-5.

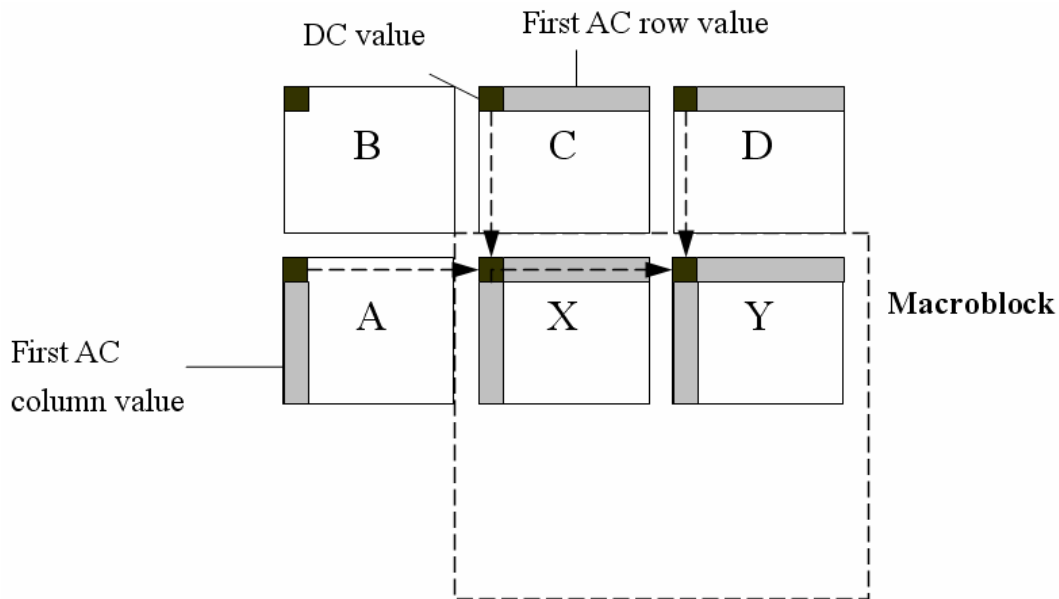


Figure 2-5 AC/DC prediction process for intra coded MBs.

In Fig. 2-5, the black square in the top-left corner of each block indicates the DC coefficient. The vertical gray rectangle indicates the column AC coefficients and the horizontal gray rectangle indicates the row AC coefficients. The prediction is either from the left or the top neighboring block. The prediction direction will be selected according to the

horizontal and vertical DC gradients around the block. As in the table 2-1, the DC gradient from B to A and the DC gradient from B to C are compared. That decides the prediction direction of block X and the lowest gradient is chosen for the AC/DC prediction. In addition, if any block is outside the VOP, its DC coefficient value will be set to $2^{(\text{bits_per_pixel} + 2)}$.

Table 2-1 AC/DC prediction pseudo code

<pre> if (F_A[0][0] - F_B[0][0] < F_B[0][0] - F_C[0][0]) predict from block C else predict from block A </pre>

2.3.4 Scan

In order to build the symbols of entropy coding, we need to transform the two-dimensional matrix values into a one-dimensional vector. A scanning process is used for the 2D to 1D conversion. There are three scanning modes defined in the MPEG-4 standard. They are the zigzag scan, the alternate horizontal scan, and the alternate vertical scan and these modes are shown in Fig. 2-6. The scan that shall be used is decided by the following method. For intra blocks, if non-prediction occurs, the zigzag scan is selected for all blocks in a macroblock. Otherwise, the DC prediction direction is used to select a scan on block basis. For example, if the DC prediction direction refers to the horizontally adjacent block, the alternate-vertical scan is chosen for the current block. Otherwise, the alternate-horizontal scan is selected for the current block if DC prediction direction refers to vertically adjacent block. For all other blocks, the 8x8 blocks of transform coefficients are scanned in the “zigzag” scanning direction.

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	33	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Alternate
Horizontal scan

Alternate
Vertical scan

Zigzag scan

Figure 2-6 Alternate MPEG-4 scanning mode

2.3.5 Variable Length Coding

In many cases, the quantization truncates the frequency components to zero. Therefore encoder uses variable length coding to encode the data, and the VLC will replace the string of zero with a count of how many zeros. The MPEG-4 adopts a novel 3-D VLC to encode the video stream. Traditional VLC is a 2-D run and level coding. Where run represents the number of zero coefficients preceding a non-zero coefficient in the scan order, and level represents the absolute value of this non-zero coefficient. As for the MPEG-4, additional symbol “LAST” indicates whether this is the final nonzero coefficient in the block. The standard describes how the actual values of these data elements (coefficients, zero runs, and LAST) can be decoded from the variable length codes. Finally, Huffman coding is used for the entropy coding.

Chapter 3 Architecture Design of MPEG-4 Video Texture Coding

3.1 Texture Coding Units Overview

In this chapter, the architecture of texture coding units is described. The texture coding units include the DCT, the quantization, the inverse quantization, the inverse DCT, and the AC/DC prediction and the architecture is shown in Fig. 3-1. Before the texture coding engine, a ping-pong buffer is required to latch the data. This ping-pong buffer consists of two memories and can generate two independent addresses for reading and writing at the same time. Which memory will be selected to read or write data is controlled by texture controller. When data from the ping-pong buffer is valid, the DCT/IDCT block is used to transfer the spatial values to the frequency values. Based on the row-column decomposition technique, the DCT and IDCT share the same architecture unit to improve implementation efficiency.

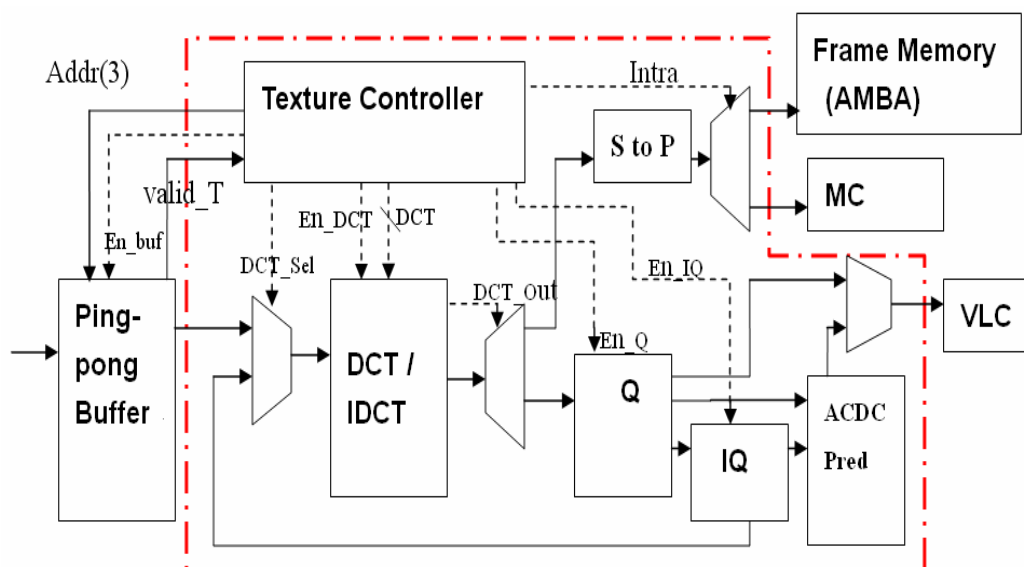


Figure 3-1 Overall architecture of MPEG-4 texture coding

Two processing paths in the texture coding architecture are shown in Fig. 3-2. One is the encoding path including the DCT, the quantization, the AC/DC prediction, and the VLC in intra coding mode. The other is the reconstructed path consisting of the inverse quantization, the IDCT, and the serial-to-parallel block. The texture controller sends out the signal “DCT” to decide which function of DCT or IDCT will be activated in the DCT/IDCT block. There are two ways to get the input data for the DCT/IDCT. One is from the ping-pong buffer and the data is used for the DCT procedure; the other is from the inverse quantization block and the data is used for the IDCT procedure. A de-multiplexer controlled by the signal “DCT_out” decides the path of the output data from the DCT/IDCT block. The output data of IDCT are pixel-by-pixel, but the reconstructed frame data delivering through AMBA bus are four pixels per time. Therefore, a serial-to-parallel block is required. When the data come out, the flag “intra” selects the destination which is either the frame memory (through AMBA) or the motion compensation unit.

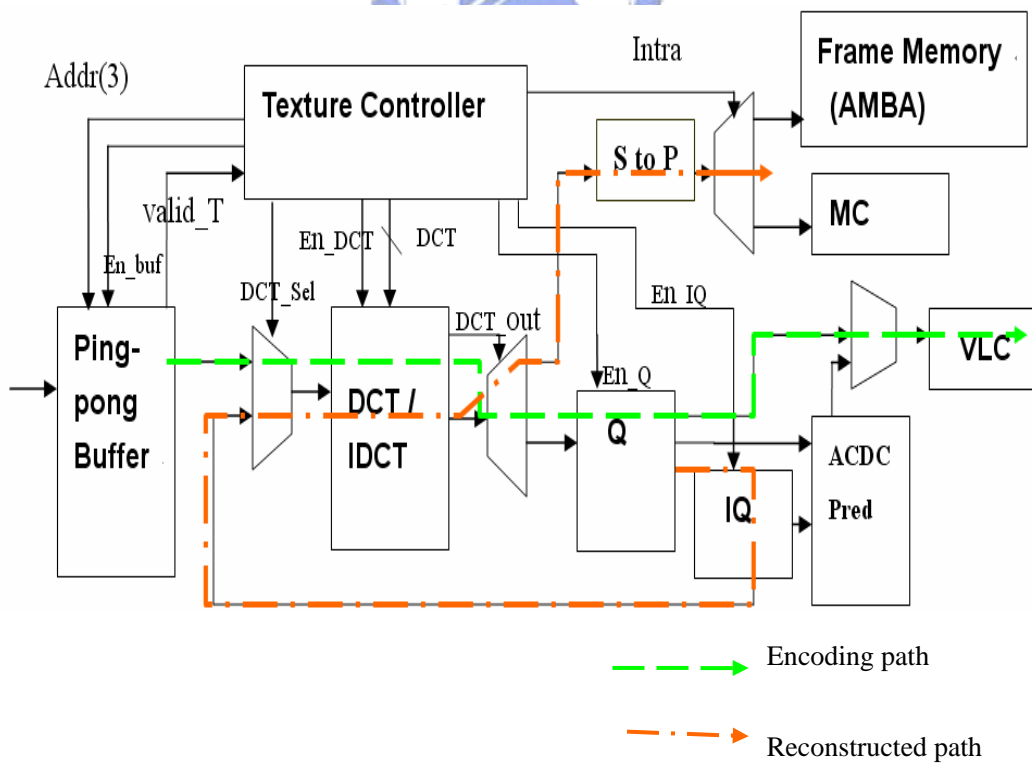
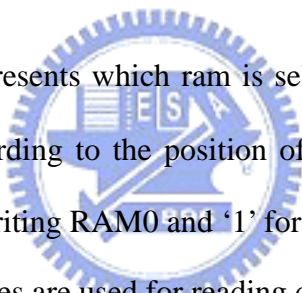


Figure 3-2 Architecture of MPEG-4 texture coding data path

In the following sections, the ping-pong buffer, the DCT/IDCT architecture, the quantization, the inverse quantization, and the AC/DC prediction block are described in more detail.

3.2 Ping-pong buffer unit

In the ping-pong buffer, there are two ram memories with the same size and each memory is 3456 bits ($64(\text{pixels/block}) \times 9(\text{bits/pixel}) \times 6(\text{blocks}) = 3456 \text{ bits}$). The pin definition of the ping-pong buffer is described as follow. The architecture of the ping-pong buffer is shown in Fig. 3-3.

- 
- WR_RAM_Sel : This flag represents which ram is selected for writing data, and the flag is decided according to the position of the current MB. The WR_RAM_Sel sets '0' for writing RAM0 and '1' for writing RAM1.
- RAddr : These addresses are used for reading data. RAddr [3] decides which ram will be read, and RAddr [2:0] determines the index of the sub-block (including Y1, Y2, Y3, Y4, Cb, and Cr) in this ram.
- Intra : This flag reveals the coding mode. In intra coding mode, input data are got from the frame memory through AMBA bus. In inter coding mode, input data are the errors from the motion compensation.
- WAddr : These address are used for writing data. WAddr [6:4] determine the number of the sub-block in the macroblock, and WAddr [3:0] are the data addresses of the sub-block.

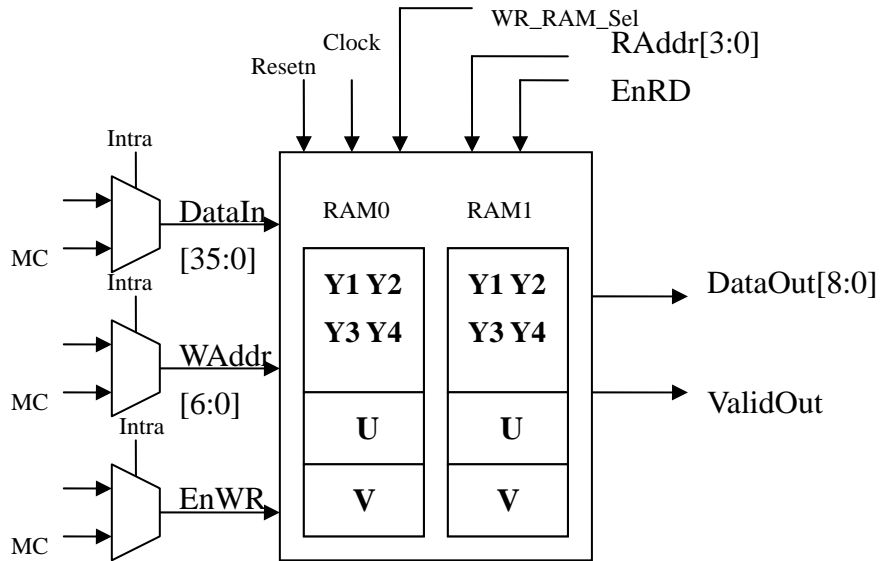


Figure 3-3 Architecture of ping-pong buffer

3.3 DCT/IDCT Architecture

A popular approach for the implementation of the 2-D DCT/IDCT is the row-column decomposition method [8], [9]. The 2-D transformation is computed by applying the 1-D DCT/IDCT by rows and, columns to reduce the complexity and hardware cost. A transpose memory is necessary to record the data between the two 1-D DCT / IDCT and the coefficients from column-by-column to row-by-row. We implement the DCT/IDCT architecture based on Weiping Li's algorithm >. The basic computation performed by the DCT/IDCT is the evaluation of the $N \times N$ matrix by the products of $N \times 1$ vectors. The computation of the product of triple matrix are, $Z=AXA^T$, for the DCT and $Z=A^T XA$ for the IDCT [12], where A is an 8×8 matrix shown in equation (3-1). Even rows of A are even-symmetric and odd rows of A are odd-symmetric. Thus, we can separate this matrix into the even and odd rows by exploiting the symmetry in the row of A.

$$A = \begin{bmatrix} a & a & a & a & a & a & a & a \\ b & d & e & g & -g & -e & -d & -b \\ c & f & -f & -c & -c & -f & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & e \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & f & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix}, \text{ where } \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos \frac{\pi}{4} \\ \cos \frac{\pi}{16} \\ \cos \frac{\pi}{8} \\ \cos \frac{3\pi}{16} \\ \cos \frac{5\pi}{16} \\ \cos \frac{3\pi}{8} \\ \cos \frac{7\pi}{16} \end{bmatrix} \quad (3-1)$$

The 2-D DCT/IDCT architecture of row-column decomposition is shown in Fig. 3-4. The 2-D DCT/IDCT transform is typically separated into two 1-D DCT/IDCT transformation to reduce the area cost and complexity. The row-column decomposition technique is implemented in two ways : DRCD (direct row-column decomposition) and MRCD (multiplexed row-column decomposition) shown in Fig. 3-4(a) and Fig.3-4(b), respectively. The DRCD architecture consist two 1-D DCT/IDCT units and one transpose memory. Compared with the MRCD, it needs lager hardware cost but fewer latency. The MRCD architecture requires only one 1-D DCT/IDCT unit and one transpose memory. It uses the multiplexer and the de-multiplexer to determine the processing path. Because the row and column share the same 1-D DCT/IDCT unit, next data are not allowed to input when 1-D DCT/IDCT unit is working. It results in the longer latency and the fewer throughput rate. But the MRCD needs smaller area than the DRCD.

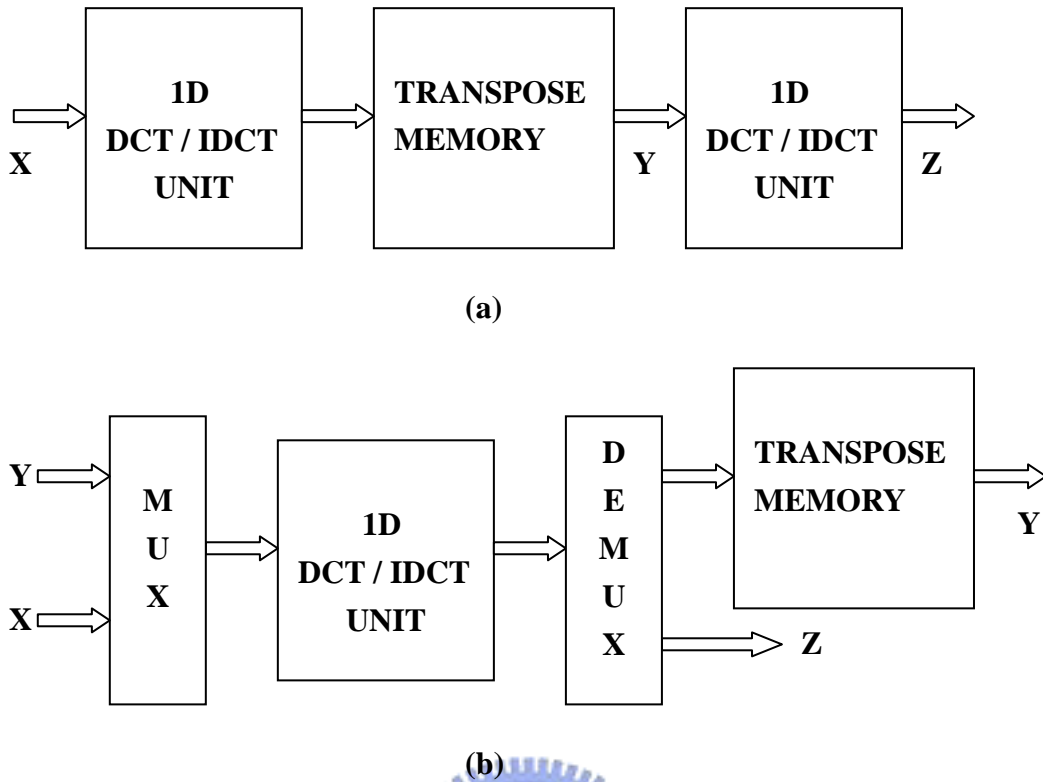


Figure 3-4 2-D DCT/IDCT architecture of row-column decomposition

The 1-D DCT/IDCT architecture consists of many processing units as shown in Fig. 3-5. Both DCT and IDCT have five-stage pipeline architecture. It consists the following blocks.

I. Serial to parallel

A serial-to-parallel unit is needed because the DCT/IDCT requires the 8 pixels input in parallel.

II. Pre-processor

The pre-processing unit of the DCT/IDCT will produce a set of 8 new values according to (3-2). These new values could be computed by the additions and subtractions of combinations of the input pixels.

III. Multiplier-adder

The multiplier-adder based architecture is used to accumulate the eq. (3-3) to the eq. (3-5). As shown in Fig. 3-6, the multiplier-adder unit consists of four multipliers and three adders.

IV. Post-processor

The post-processing unit will produce 8 DCT/IDCT coefficients according to eq. (3-6)

V. Parallel to serial

Finally, a parallel-to-serial unit is used to generate the output coefficients in serial.

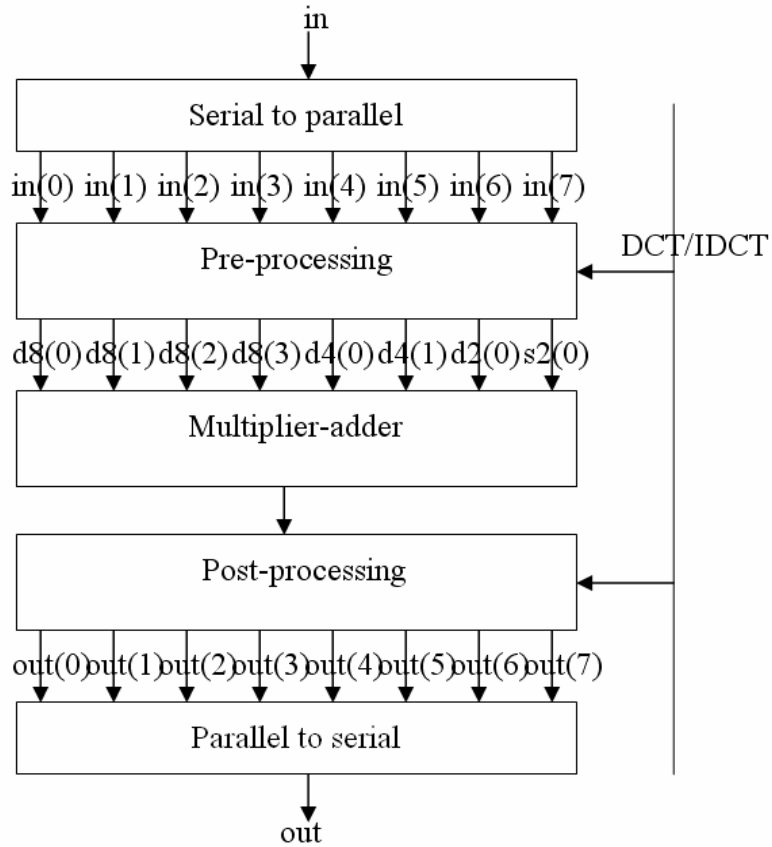


Figure 3-5 Architecture of 1D DCT/IDCT unit

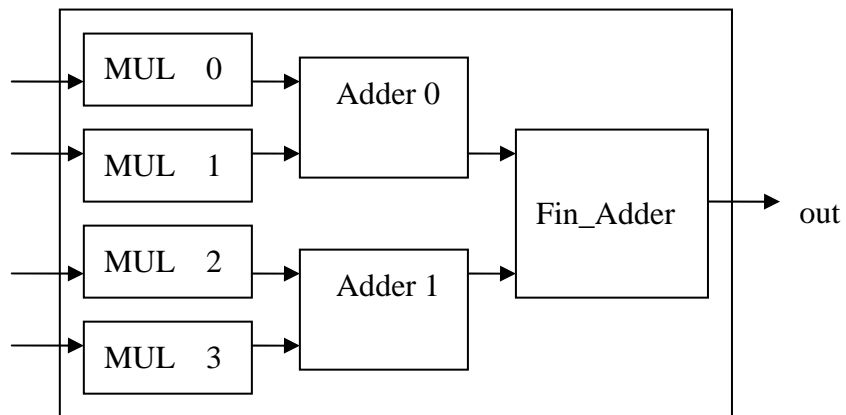


Figure 3-6 Architecture of multiplier-adder unit

IF DCT

$$\begin{cases} d_8(0) = in(0) - in(7) \\ d_8(1) = in(1) - in(6) \\ d_8(2) = in(4) - in(3) \\ d_8(3) = in(2) - in(5) \\ d_4(0) = (in(0) + in(7)) - (in(3) + in(4)) \\ d_4(1) = (in(1) + in(6)) - (in(2) + in(5)) \\ d_2(0) = (in(0) + in(7) + in(3) + in(4)) - (in(1) + in(6) + in(2) + in(5)) \\ S_2(0) = (in(0) + in(7) + in(3) + in(4)) + (in(1) + in(6) + in(2) + in(5)) \end{cases}$$

IF IDCT

$$\begin{cases} d_8(0) = in(0) \\ d_8(1) = in(3) \\ d_8(2) = -in(7) \\ d_8(3) = in(5) \\ d_4(0) = in(2) \\ d_4(1) = in(6) \\ d_2(0) = in(4) \\ S_2(0) = in(0) \end{cases} \quad (3-2)$$

$$\begin{cases} m(1) = C_8(0) * d_8(0) + C_8(1) * d_8(1) + C_8(2) * d_8(2) + C_8(3) * d_8(3) \\ m(3) = -C_8(0) * d_8(3) + C_8(1) * d_8(0) + C_8(2) * d_8(1) + C_8(3) * d_8(2) \\ m(7) = -C_8(0) * d_8(2) - C_8(1) * d_8(3) + C_8(2) * d_8(0) + C_8(3) * d_8(1) \\ m(5) = -C_8(0) * d_8(1) - C_8(1) * d_8(2) - C_8(2) * d_8(3) + C_8(3) * d_8(0) \end{cases} \quad (3-3)$$

$$\begin{cases} m(2) = C_4(0) * d_4(0) + C_4(1) * d_4(1) \\ m(6) = -C_4(0) * d_4(1) + C_4(1) * d_4(0) \end{cases} \quad (3-4)$$

$$\begin{cases} m(4) = C_2(0) * d_2(0) \\ m(0) = C_2(0) * S_2(0) \end{cases} \quad (3-5)$$

$$C_N(i) = \sqrt{\frac{2}{8}} \cos\left(\frac{\pi * 3^i}{2N}\right) \quad (3-6)$$

IF DCT

$$\begin{cases} out(0) = m(0) \\ out(1) = m(1) \\ out(2) = m(2) \\ out(3) = m(3) \\ out(4) = m(4) \\ out(5) = m(5) \\ out(6) = m(6) \\ out(7) = -m(7) \end{cases}$$

IF IDCT

$$\begin{cases} out(0) = m(1) + m(2) + m(4) + m(0) \\ out(1) = m(3) + m(6) - m(4) + m(0) \\ out(2) = m(5) - m(6) - m(4) + m(0) \\ out(3) = -m(7) - m(2) + m(4) + m(0) \\ out(4) = m(7) - m(2) + m(4) + m(0) \\ out(5) = -m(5) - m(6) - m(4) + m(0) \\ out(6) = -m(3) + m(6) - m(4) + m(0) \\ out(7) = -m(1) + m(2) + m(4) + m(0) \end{cases} \quad (3-7)$$

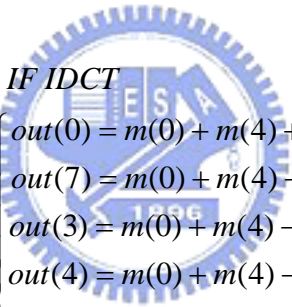
Because the $d_2(0)$ and $S_2(0)$ in the eq.3-2 have the similar items except for the operator adder or subtraction, we can rewrite $d_2(0)$ and $S_2(0)$ in the eq.3-8 to reduce the adders in the

hardware implementation. Due to the modification of eq.3-2, rewording the $m(4)$ and $m(0)$ in the eq.3-5 are necessary when the process is in the DCT mode. The new equation is described in (3-9). The numbers of the adders in the pre-processor are 12.

$$\begin{aligned} d'_2(0) &= in(0) + in(7) + in(3) + in(4) \\ S'_2(0) &= in(1) + in(6) + in(2) + in(5) \end{aligned} \quad (3-8)$$

$$\begin{cases} m(4) = C_2(0) * (d'_2(0) - S'_2(0)) \\ m(0) = C_2(0) * (d'_2(0) + S'_2(0)) \end{cases} \quad (3-9)$$

We can rearrange (3-7) to get fewer adders as shown in (3-10). The adders in the post-processor are 14.

<p><i>IF DCT</i></p> $\left\{ \begin{array}{l} out(0) = m(0) \\ out(1) = m(1) \\ out(2) = m(2) \\ out(3) = m(3) \\ out(4) = m(4) \\ out(5) = m(5) \\ out(6) = m(6) \\ out(7) = -m(7) \end{array} \right.$	 <p><i>IF IDCT</i></p> $\left\{ \begin{array}{l} out(0) = m(0) + m(4) + m(2) + m(1) \\ out(7) = m(0) + m(4) + m(2) - m(1) \\ out(3) = m(0) + m(4) - m(2) - m(7) \\ out(4) = m(0) + m(4) - m(2) + m(7) \\ out(1) = m(0) - m(4) + m(6) + m(3) \\ out(6) = m(0) - m(4) + m(6) - m(3) \\ out(2) = m(0) - m(4) - m(6) + m(5) \\ out(5) = m(0) - m(4) - m(6) - m(5) \end{array} \right. \quad (3-10)$
---	---

The complexity of our 1-D DCT/IDCT algorithm is depicted in Table 3-1. We need four multipliers and three adders in the multiplier-adder module. Twelve and fourteen adders are required in the pre-processor and the post-processor, respectively. One adder is used for rounding after the parallel to serial block. Total number of the adders in our proposed 1-D DCT/IDCT algorithm is 30.

Table 3-1 Complexity of the proposed 1-D DCT/IDCT algorithm

	Our proposed
Multipliers	4
Adders	12+3+14+1=30

As shown in Table 3-2, our design uses fewer multipliers to achieve the 1-D DCT/IDCT architecture while comparing with previous work. The design of [11] is also based on the weiping Li's algorithm and needs 7 multipliers to implement a 1-D DCT/IDCT. The design of [13] is required 9 multipliers and 21 adders to achieve.

Table 3-2 complexity of different 1D DCT/IDCT design

Algorithm	Cheng's[13]	Bousselmi's[11]	Our proposed
Multipliers	9	7	4
Adders	21	31	30

Fig. 3-7 shows the read/write action of the DCT/IDCT transpose memory [14]. For the row-column decomposition of the 2D DCT/IDCT, the coefficients of the 1-D DCT/IDCT are written into the transpose memory row-by-row in sequence (0, 1, 2, 3, 4, 5, 6, 7, 8, ...). As the coefficients of 1-D DCT/IDCT are written to the address 49, the data in the first column can be prepared to read. After that, the data in the transpose memory will be read column by column, (0, 8, 16, 24, 32, 40, 48, 56, 1,...). As shown in Fig. 3-7 (b), the data written to the address 56 is ready to be read after 8 cycles. At next time slice, other coefficients of the 1-D DCT/IDCT will be written column by column and read row by row. Therefore, reading and writing data in transpose memory can be achieved at the same time under this structure.

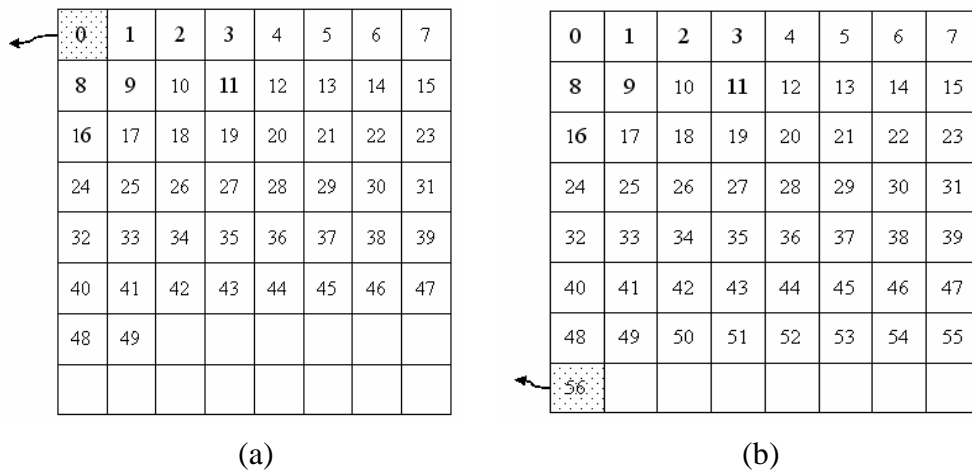


Figure 3-7 Read/write action of DCT/IDCT transpose memory

A multiplier-adder based DCT/IDCT architecture is shown in Fig. 3-7. In Fig. 3-7, there are five quantization error sources: 1. The quantization of the coefficients for the row-wise and the column-wise transform (Coeff1 and Coeff2). 2. The wordlength reduction for the outputs of the first and the second multipliers (Adder1 and Adder2). 3. The output of the limiter for the row-wise transform (1D_Out). The most suitable way for deciding the minimum wordlength of the Coeff1, the Coeff2, and the 1D_Out is to compute the overall mean square error. The peak mean error, and the overall mean error are important to determine the minimum wordlength of Adder1 and Adder2 [15]. The optimum wordlength of these five terms in our work are shown in Table 3-3. The IDCT precision of this module meets IEEE IDCT precision standard and the implementation results are shown in chapter 4.

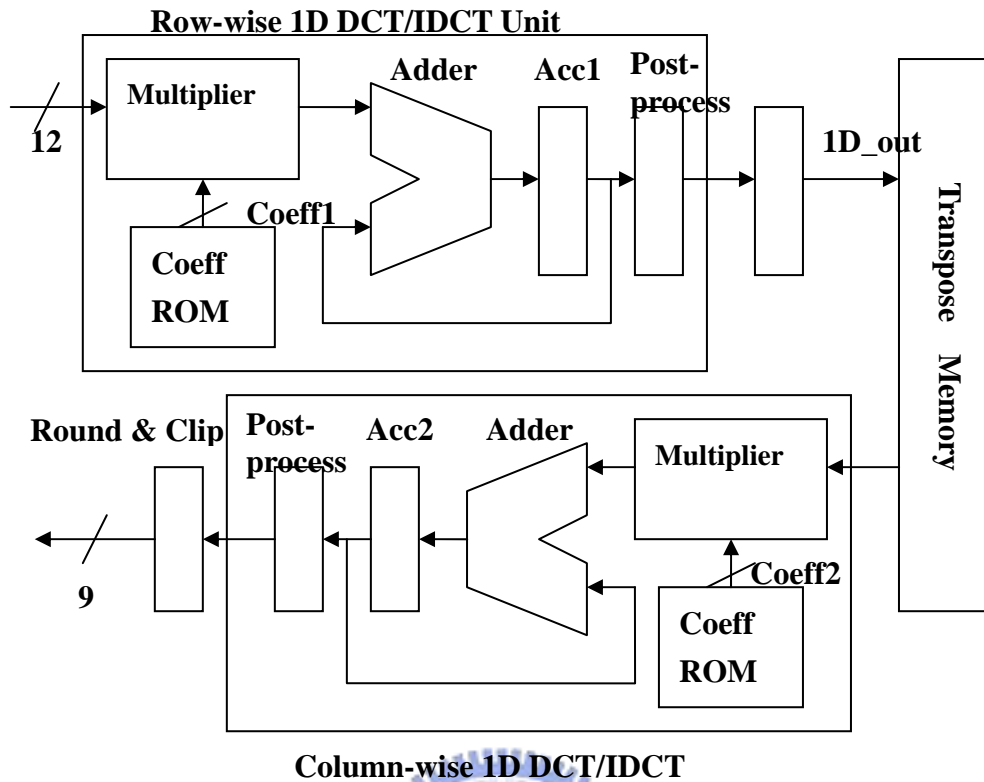


Figure 3-8 Block diagram of a multiplier-adder based 2-D DCT/IDCT

Table 3-3 The optimized wordlength for the our 2-D DCT/IDCT architecture

	Optimized Word Length
Coeff1	13
Acc1	21
1D_Out	16
Coeff2	12
Acc2	20

In order to reduce the hardware cost of the architecture of the DCT/IDCT, rounding is required after a multiplication. In our architecture, rounding is used to truncate the output data of each 1D DCT/IDCT module. We adopt the true rounding method to improve the IDCT precision. For the $n \times n$ multiplication, true rounding requires adding a 1 at the n th least significant bit of the product and truncates the least significant n bits of the sum. This process is illustrated via a dot diagram for a six by six multiplier on Fig.3-9[16].

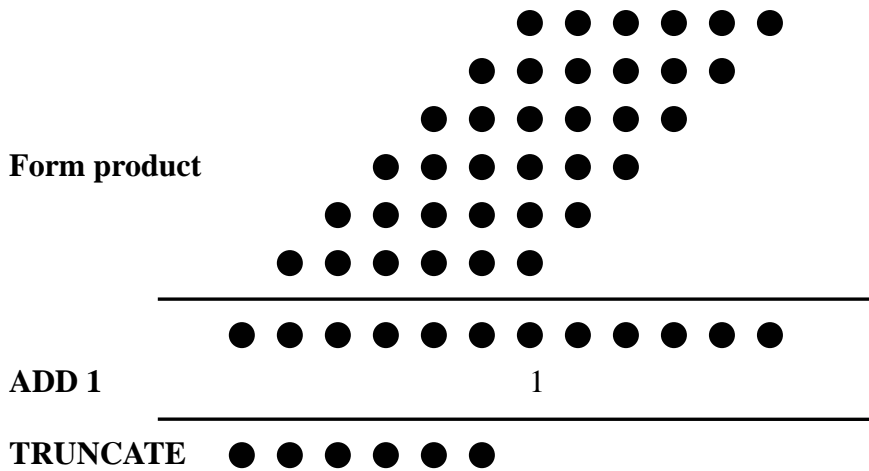


Figure 3-9 Rounded Multiplication Dot Diagram

The core characteristics of the DCT/IDCT architecture have been summarized in Table 3-4.

Table 3-4 Core characteristics of the DCT/IDCT architecture

Inputs	9 bits(DCT), 12 bits(IDCT)
Outputs	12 bits(DCT), 9 bits(IDCT)
Internal wordlength	16 bits
Technology	0.18-um CMOS
No. of transistors	168,244
Clock size	70 MHz
Mode Selection	DCT or IDCT
Block size	8 x 8
Accuracy	IEEE std. 1180-1990

3.4 Quantization and inverse Quantization Design

Quantization applied to the transform coefficients can be viewed as division followed by the integer truncation. The division could be regarded as multiplying the reciprocal number. Consequently, the hardware architecture of the divider could be achieved by a multiplier and a

shifter. In order to meet the quantized values defined in the MPEG-4 standard, the quantization tables have to be established. When building up the N-bit quantization table, the reciprocal of the quantization step size must be set to $2^N / x + 1$ (where x is the quantization step size). For example, while the step size is 4, the 16-bit binary data is $(0100000000000001)_{bin}$. After multiplying the binary value in the quantization table, the shift for truncating data to integer is required. The MPEG-4 standard adopts a non-linear scaler for DC coefficients of the DCT blocks specified in Table 3-5

Table 3-5 Non linear scaler for DC coefficients of DCT blocks, expressed in terms of relation with quantizer_scale

Component: Type	dc_scaler for quantizer_scale range			
	1 ~ 4	5 ~ 8	9 ~ 24	>=25
Luminance: Type1	8	$2 \times \text{quantizer_scale}$	$\text{quantizer_scale} + 8$	$2 \times \text{quantizer_scale} - 16$
Chrominance: Type2	8	$(\text{quantizer_scale} + 13) / 2$		$\text{quantizer_scale} - 6$

The architecture of the quantization has to select a proper quantization table according to the quantized type of the current data. The type definitions are based on the luminance or the chrominance, the AC value or the DC value, and the inter block or the intra block. The flowchart of the quantization is shown in Fig. 3-10. First, the coding mode of the block is judged by the intra signal. In the intra coding mode, different quantization tables are provided for the luminance and the chrominance of the DC values. In some cases, H.263 quantizer guarantees that all coefficients equal to zero. As shown in Table 3-6, the quantized value is set to zero if the absolute value of the DCT coefficient is less than $2.5 \times \text{quantization parameter}$ in the inter coding mode. In the intra coding mode, the threshold is set to $2 \times \text{quantization parameter}$.

Table 3-6 H.263 quantizer to guarantee all coefficients equal zero.

H.263 Quantizer	
INTER mode for AC/DC coefficients	$ F(u, v) - QP/2 < 2*QP$ to guarantee all $coeff(u,v) = 0$
INTRA mode for AC coefficients	$ F(u, v) < 2*QP$ to guarantee all $coeff(u,v) = 0$

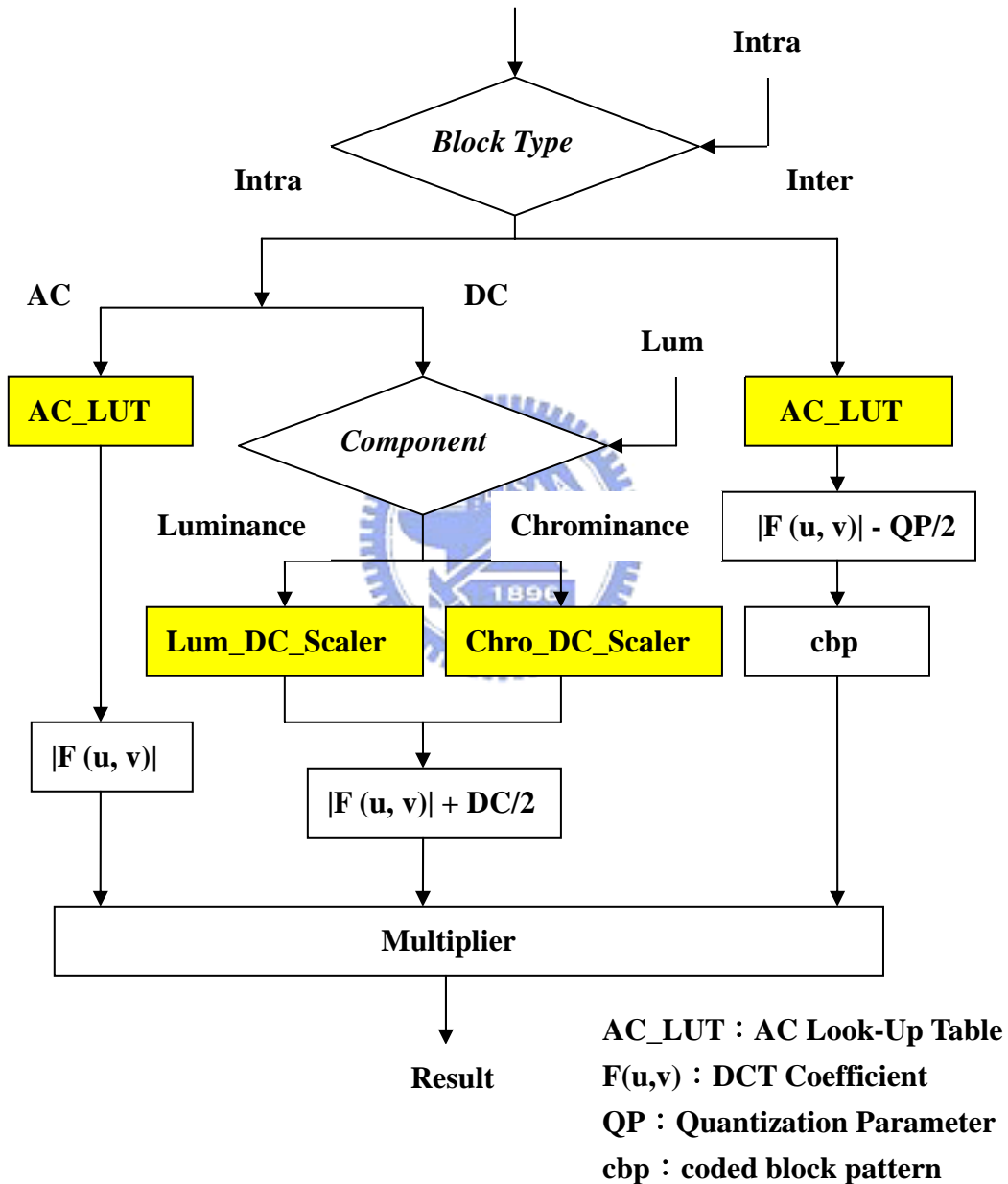


Figure 3-10 Flowchart of the quantization architecture

Each block has one bit to represent a coded/no coded status of it. In the inter coding mode,

the value of the cbp is set to '1' if any coefficient in the block is not zero. In the intra coding mode, cbp is necessary to accumulate the absolutes of coefficients in this block. As shown in Fig. 3-11, when one of the grey pixels in the top left corner of the block is not zero, or the sum of all the absolutes of coefficients is greater than 2, which implies that this block needs to be encoded, the value of this cbp is set to '1'.

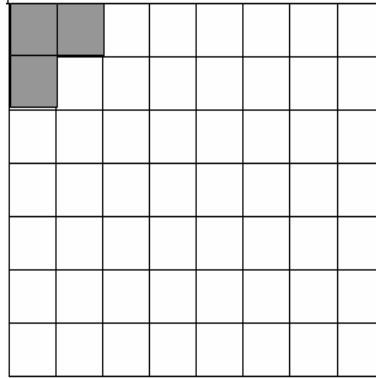


Figure 3-11 coded block pattern in intra coding mode

The inverse quantizer has similar architecture with the quantizer, except the data with fewer bits in the look up tables. There is the mismatch due to exact values (integer + 1/2) of the IDCT output, so the reconstruction level of the quantizer must be designed to alleviate this problem. A simple way to solve this mismatch problem is to avoid even values for the reconstruction levels. The uniform quantizer adopted in CCITT H.261-1990, Video Codec for Audiovisual Services at px64 kbit/s, the reconstruction levels (REC) are defined as follows[17] :

$$\begin{aligned}
 REC &= QUANT * (2 * LEVEL + 1); & LEVEL > 0 \\
 REC &= QUANT * (2 * LEVEL - 1); & LEVEL < 0 & \quad QUANT = "odd" \\
 \\
 REC &= QUANT * (2 * LEVEL + 1) - 1; & LEVEL > 0 \\
 REC &= QUANT * (2 * LEVEL - 1) + 1; & LEVEL < 0 & \quad QUANT = "even" \\
 REC &= 0; & LEVEL = 0,
 \end{aligned}
 \tag{3-11}$$

where QUANT ranges from 1 to 31 whose value corresponds to half of the step size.

Fig.3-12 shows the flowchart of the inverse quantizer. First, to judge the current block is intra or inter is required. If the current block is the intra block, it implies the DC values should be got from multiplying the coefficients by the dc_scaler value. For other coefficients, the dequantized values will be obtained from the eq. 3-11.

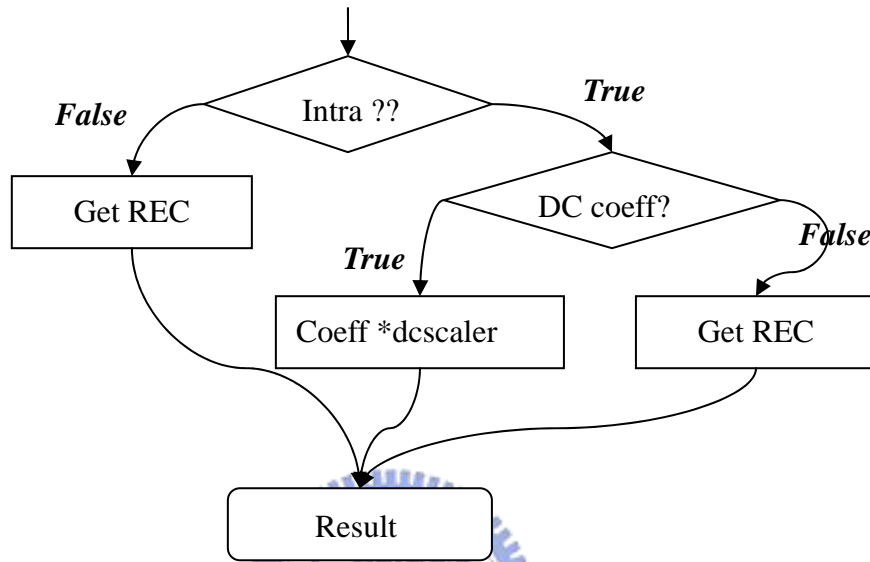


Figure 3-12 flowchart of inverse quantizer

3.5 AC/DC prediction design

In order to perform the AC/DC prediction, a large memory to store the information of the frame is necessary and the Y component storage is stored as in Fig.3-13. After the prediction of a block is complete, it is necessary to update the coefficient. For instance, while the prediction of block 0 is accomplished, the DC coefficient is copied to that of block B and D. The top AC coefficients are copied to the top locations of the block D and the left AC coefficients are copied to the left locations of the block B. The decision of the prediction direction is based on the DC coefficient of the neighbor blocks of the current block. Let's take the block 0 as an example, the DC value of block C, B, and D are used to determine the prediction direction.

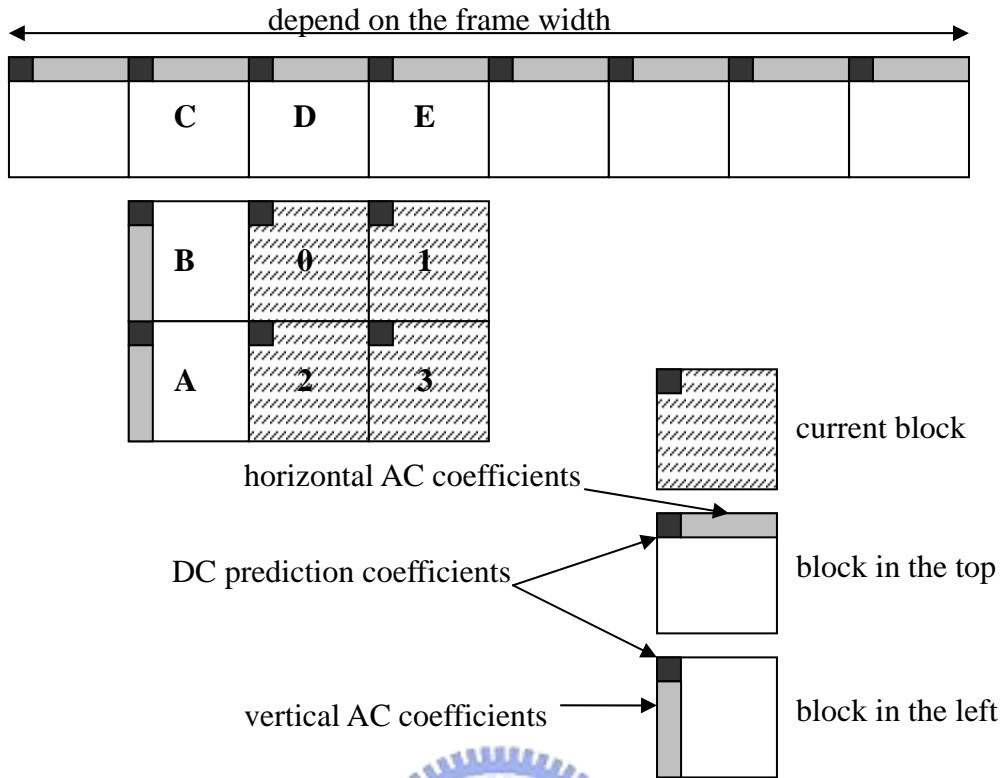


Figure 3-13 Y component storage for AC/DC prediction

The architecture of K.Suh is adopted to implement the AC/DC predictor [18]. Fig. 3-14 shows the structure of the memory to store the prediction data for CIF 352x288 resolutions. The horizontal memory stores the DC and horizontal AC coefficients and its size depends on the frame width. The vertical memory stores the vertical DC and AC coefficients. LT_DC_VALUE memory stores the top left value to be predicted. The top left value is replaced by the top value of each 8x8 block when a block read top value according to the table 3-8.

742x12 bits		
horizontal_memory (352(Y) + 352(Cr, Cb) x12 = 704x12 bits	LT_DC_VALUE 6x12 bits	vertical_memory 32x12 bits

Figure 3-14 Structure of prediction memory

Table 3-7 shows that the left top DC values transfer in each prediction block. For example, when current block index is 0, the left top DC value is read from index 1 of LT_DC_VALUE memory for prediction. The top value of block index 0 is stored to the LT_DC_VALUE memory with index 0 and this data is the left top DC value of the block 1.

Table 3-7 Storing left top value and reading for each block index

Block index	0	1	2	3	4	5
Storing LT_value	0	1	2	3	4	5
LT memory read	1	0	3	2	4	5

To perform the AC/DC prediction, the division is required to implement the normalization for the DC coefficient. In order to reduce the hardware cost of the AC/DC predictor, the quantizer will be used to normalize the DC coefficient. The interleaved DCT/IDCT scheduling (IDIS) [14] is adopted and modified to meet our request. The scheduling for the DCT/IDCT, the Q (quantization), the IQ (inverse quantization) units are shown in Fig. 3-15. We can use the idle time of the quantizer unit to perform the division of the DC coefficient and fetch the AC coefficients from the prediction memory.

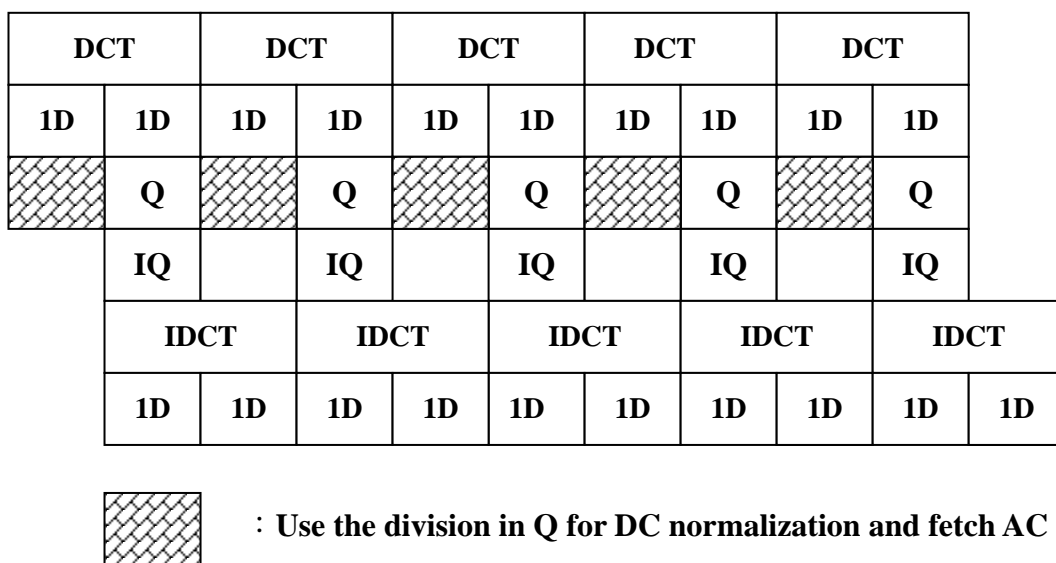


Figure 3-15 Timing diagram of interleaved DCT/IDCT scheduling

The architecture of the AC/DC prediction is depicted in Fig. 3-16. The prediction memory stores the AC/DC prediction coefficients of the past and these coefficients are used to predict. The direction manager decides the prediction direction by comparing the DC gradients. The DC normalization normalizes the boundary DC values. The state machine generates the irregular address to access the prediction memory and the register file stores the prediction values and the quantized values.

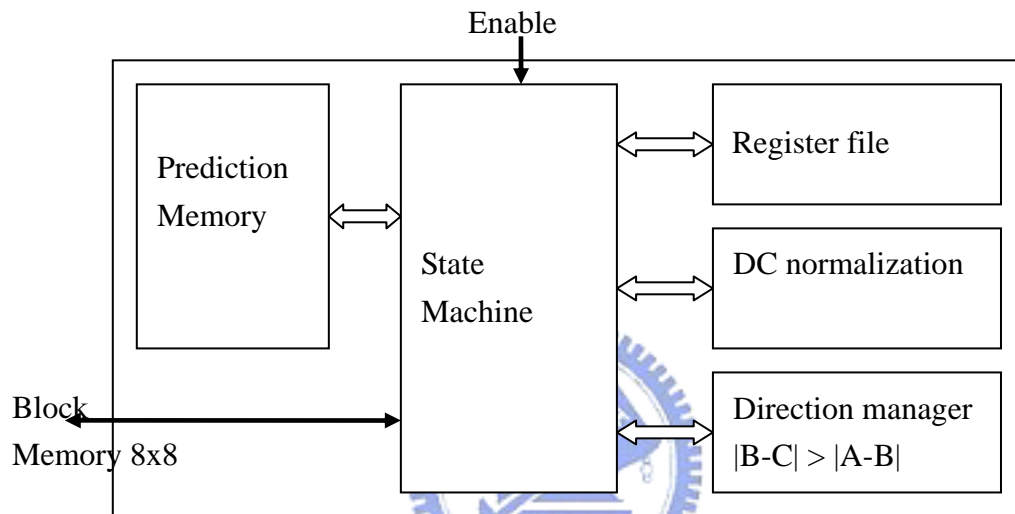
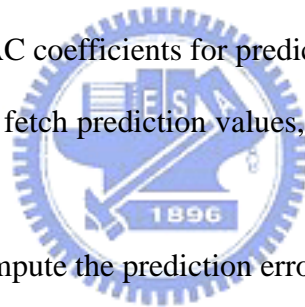


Figure 3-16 Architecture of AC/DC prediction

It doesn't guarantee that the data after AC/DC prediction are better than the data without prediction. Consequently, the coded block patterns in intra coding mode are determined by the comparison between the data after and before prediction. When the summation of the absolute value after prediction is larger than without prediction, the smaller values (quantized DCT coefficients) will be adopted to determine the coded block patterns and vice versa. In the inter coding mode, if any AC coefficient in the 8x8 block is non-zero, it implies that this block is necessary to encode. Then, the coded block pattern is set to '1'. The Finite State Machine of the AC/DC prediction is depicted in Fig. 3-17. Each state of this finite state machine is described as follows.

- ✧ IDLE: do nothing in this state, and stay until the DCT coefficient is valid in intra coding mode.
- ✧ Read LT DC : read the left top DC value from the LT_DC_VALUE memory. If the current block is a boundary block, the DC values around it may be a constant. Skipping the next states Read Top DC or Read Left DC is allowed.
- ✧ Read Top DC : get the top DC value.
- ✧ Read Left DC : get the left DC value.
- ✧ Store LT DC : store the top DC value into the LT_DC_VALUE according to Table 3-8.
- ✧ Check Gradient : determine the prediction direction and get AC coefficients from the prediction memory.
- ✧ Read Top AC : read the AC coefficients for prediction from the horizontal memory.
- ✧ Read Left AC : read the AC coefficients for prediction from the vertical memory.
- ✧ Waiting Quantized Data : fetch prediction values, and wait until the quantized values are valid.
- ✧ Store Next Prediction : compute the prediction errors and store the current quantized DCT coefficients into the prediction memory.
- ✧ ACDC To VLC : judge whether the prediction of the current macroblock is complete. Go to the ACDC cbp state when the prediction is accomplished; else go to the IDEL state then wait for the next 8x8 block.
- ✧ ACDC cbp : deliver the coded block patterns to the VLC unit then enter the IDLE state.



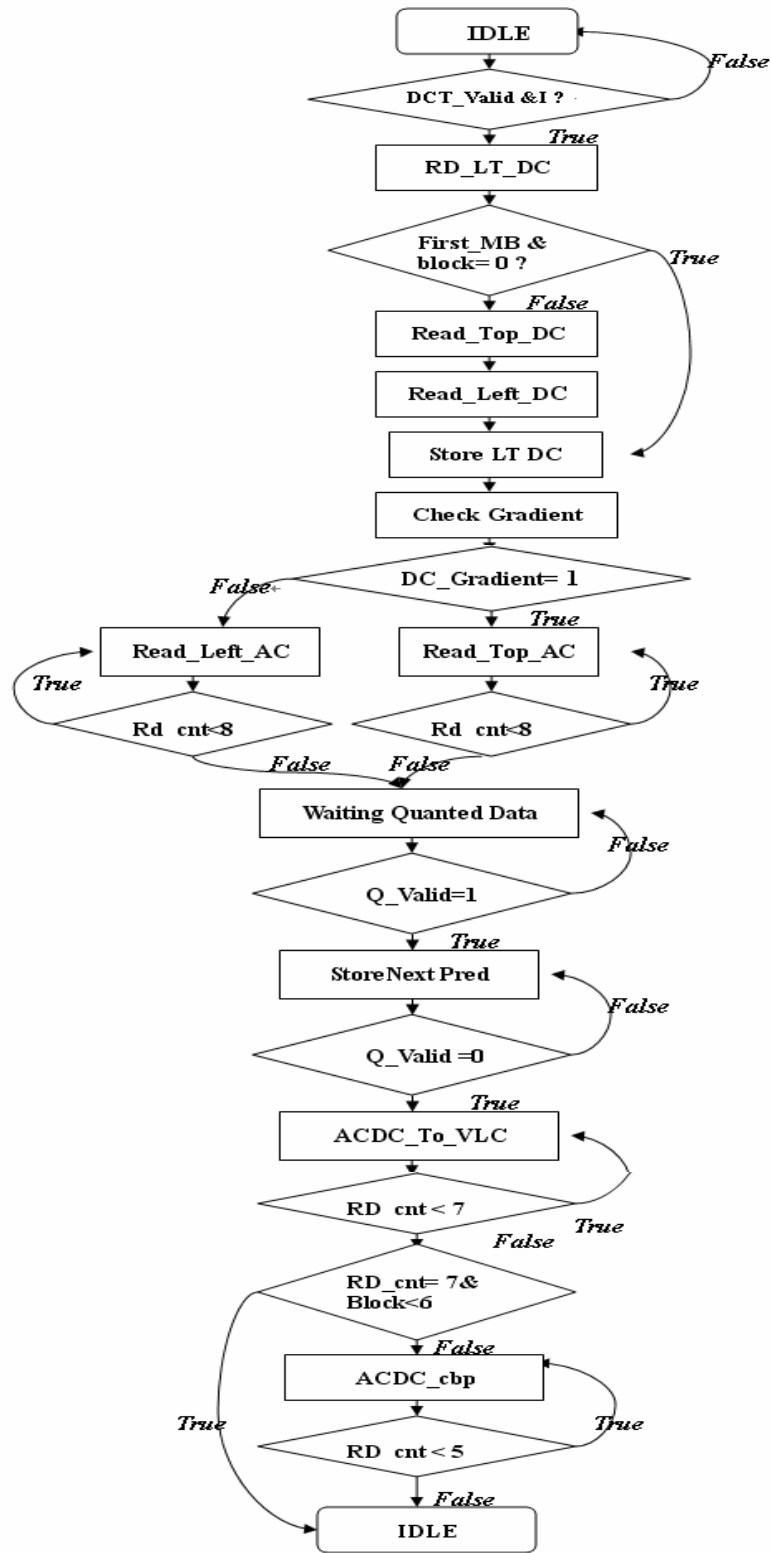


Figure 3-17 AC/DC prediction Finite State Machine

3.6 Finite State Machine of Texture Coding

The valid signals are used to represent that the values are valid between each block. Therefore, the valid signals can be taken as the judgment on the conditional branch in the finite state machine of the texture coding. There are different coding paths in the inter coding mode and the intra coding mode. In the intra coding mode, it must be through the AMBA bus to load the input data into the ping-pong buffer and write the output data to the reconstructed frame memory. In the inter coding mode, the data access is controlled by the motion estimation engine. Consequently, the AMBA access finite state machine is required only in the intra coding mode. The AMBA access finite state machine is shown in Fig. 3-18 and each state of the AMBA FSM is described as follows.

- ✧ IDLE : nothing to do in this state, and wait for enable reading or writing.
- ✧ Read Request : request to access data through AMBA bus and wait for the response of the AMBA controller. When the user of the AMBA bus is texture coding, it implies that to access data through AMBA bus is allowed.
- ✧ AMBA Get RAddr : get the address to read data.
- ✧ AMBA Get WAddr : get the address to write data.
- ✧ AMBA Read : load the data in the next macroblock into the ping-pong buffer. To improve the overall timing efficiency, a schedule of reading AMBA is essential. In order to avoid reading and writing data through AMAB bus at the same time, the data will be read in block index 3 and 6 respectively.
- ✧ AMBA Write : writing the data into the reconstructed frame memory through AMBA bus. It is required to write sixteen times each block.
- ✧ AMBA Finish : when reading or writing data is complete, send a finished flag to the texture controller then go to the IDLE state.

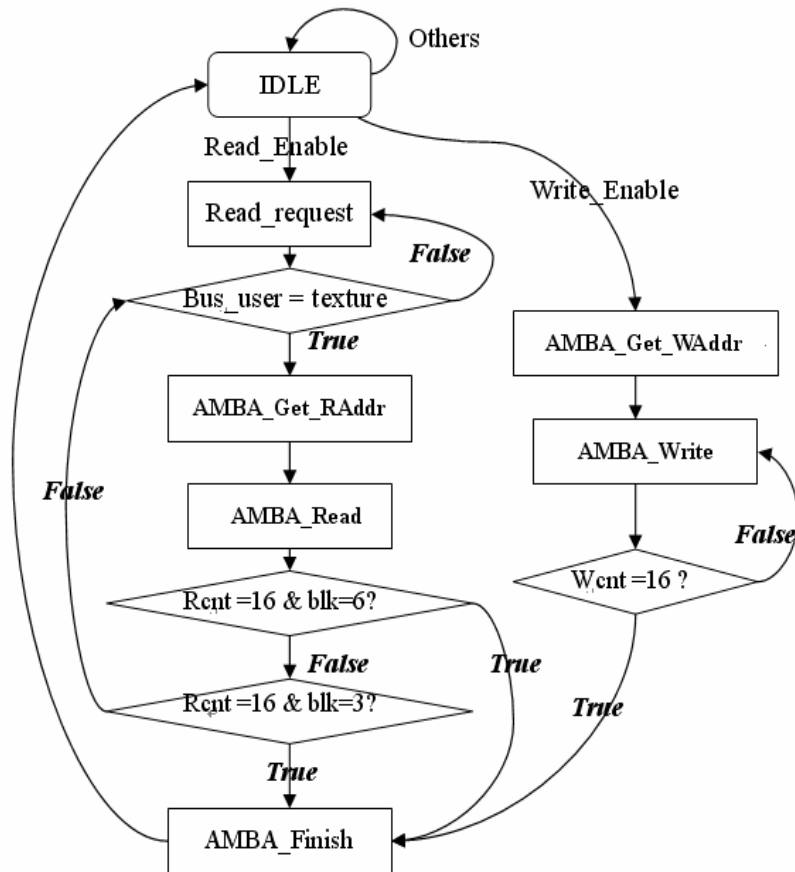


Figure 3-18 AMBA Read/Write Finite State Machine

Texture coding FSM is shown in Fig. 3-19. When the finite state machine is enabled, first step to do is to judge what kind of frame to be encoded. If the MB_I_P signal is '1', it indicates that the current frame is the intra frame; otherwise, the current block frame is the prediction frame. After encoding, the finite state machine enters the Texture_Finish state. Texture coding delivers a finished signal to the top controller, and waits for the acknowledge signal from the top controller. This finite state machine will get into the IDLE mode when the acknowledge signal is '1'.

- ✧ Texture_IDLE : texture coding engine is in the idle mode, do nothing and wait for enable signal.
- ✧ MB_Type_Check : check the current macroblock is inter or intra mode.
- ✧ Inter_Encoding : inter encoding procedure.

- ✧ Intra_Encoding : intra encoding procedure.
- ✧ Texture_Finish : texture coding for a macroblock is complete and wait for an acknowledge signal from the top controller.

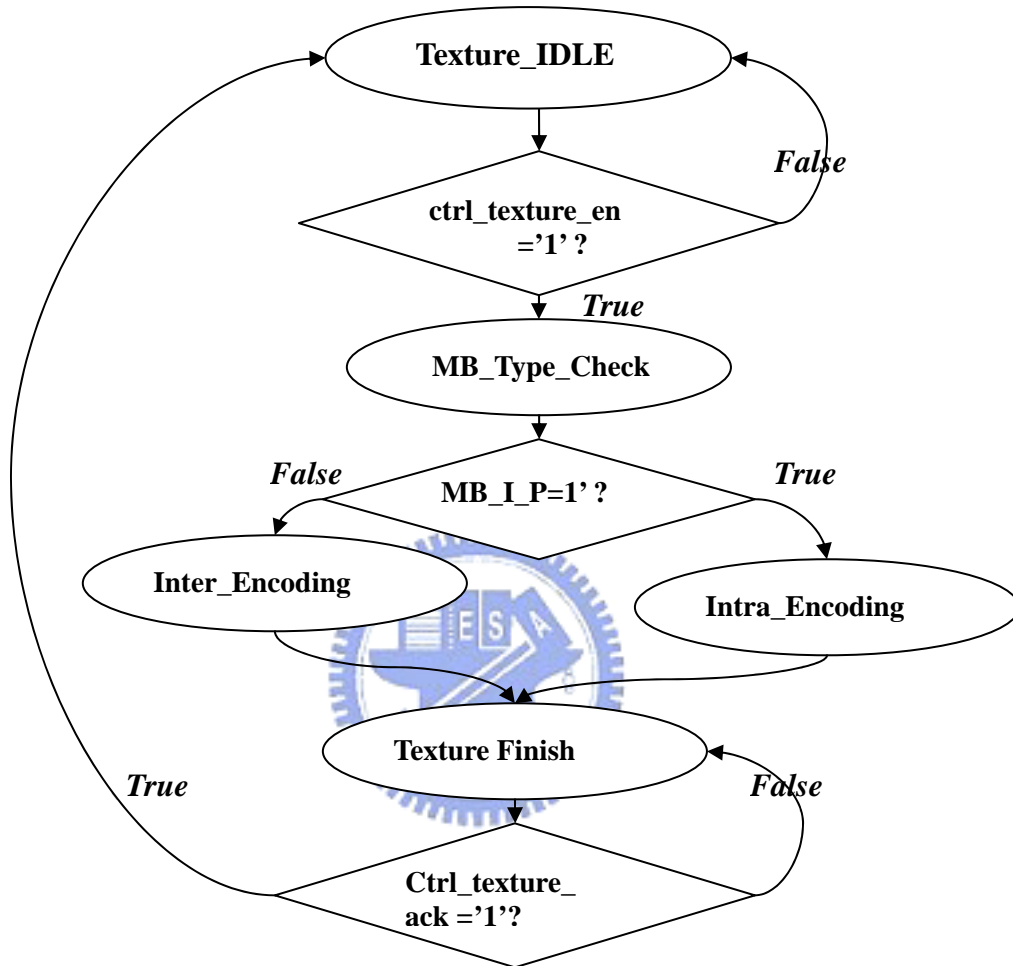


Figure 3-19 Finite State Machine of texture coding

Fig. 3-20 illustrates the intra encoding FSM. In this FSM, it is required to determine that the current macroblock is the first macroblock or not. If yes, it signifies that there is no available data in the ping-pong buffer and we have to read data from the frame memory in the first instance. When the current macroblock is not the first one, the data of the next macroblock will be read into the ping-pong buffer during the UV Block-level operation state. The Block-level operation represents the encoding operation of the 8x8 block. Because

AMBA bus is shared by the texture coding engine, the motion compensation engine and the variable length code engine, only one engine can use AMBA bus at the same time. The texture coding FSM doesn't enter the Texture_Finish state until completely writing the reconstructed values through the AMBA bus.

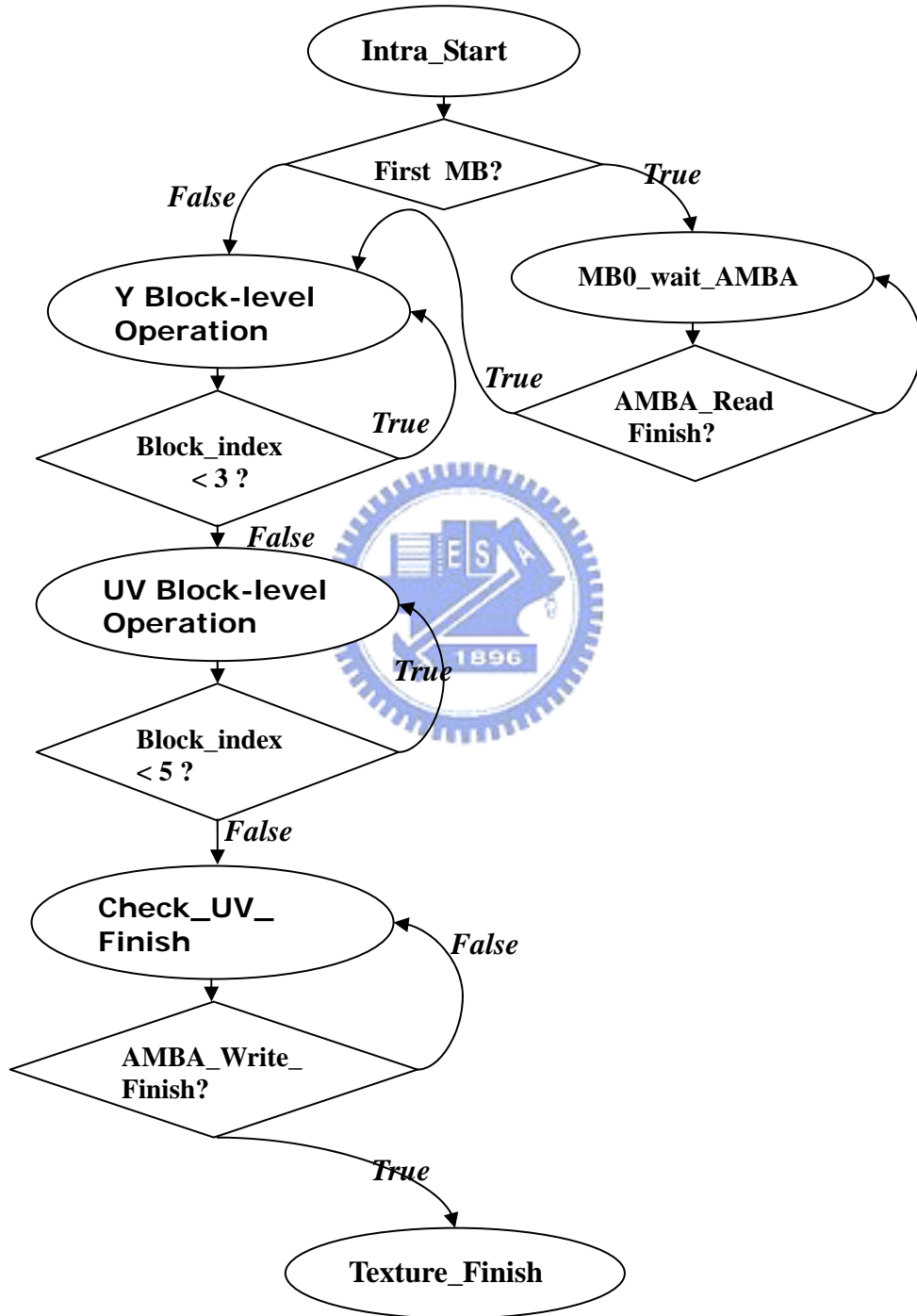


Figure 3-20 Intra encoding procedure

In inter coding mode, storing data into the ping-pong buffer is controlled by the motion estimation engine. Fig. 3-21 represents the inter encoding FSM. It is unnecessary to control the AMBA bus in this mode, so the luminance blocks and the chrominance blocks have the same procedure. The block index counts the number of the coded blocks. After the whole macroblock encoding, it is imperative to confirm that the last reconstructed value is delivered and to avoid data missing.

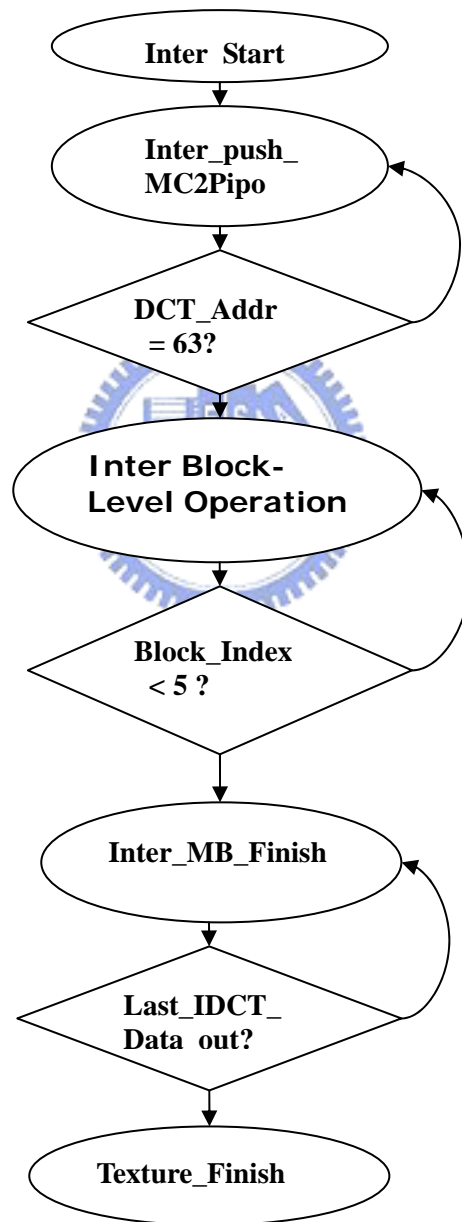


Figure 3-21 Inter encoding procedure

Basically, the block-level operation in the intra coding mode is the same as that in the inter coding mode. A flag between two blocks illustrates whether the data is valid or not. Due to the valid signals, it is extremely simple to implement the block-level operation as shown in Fig. 3-22.

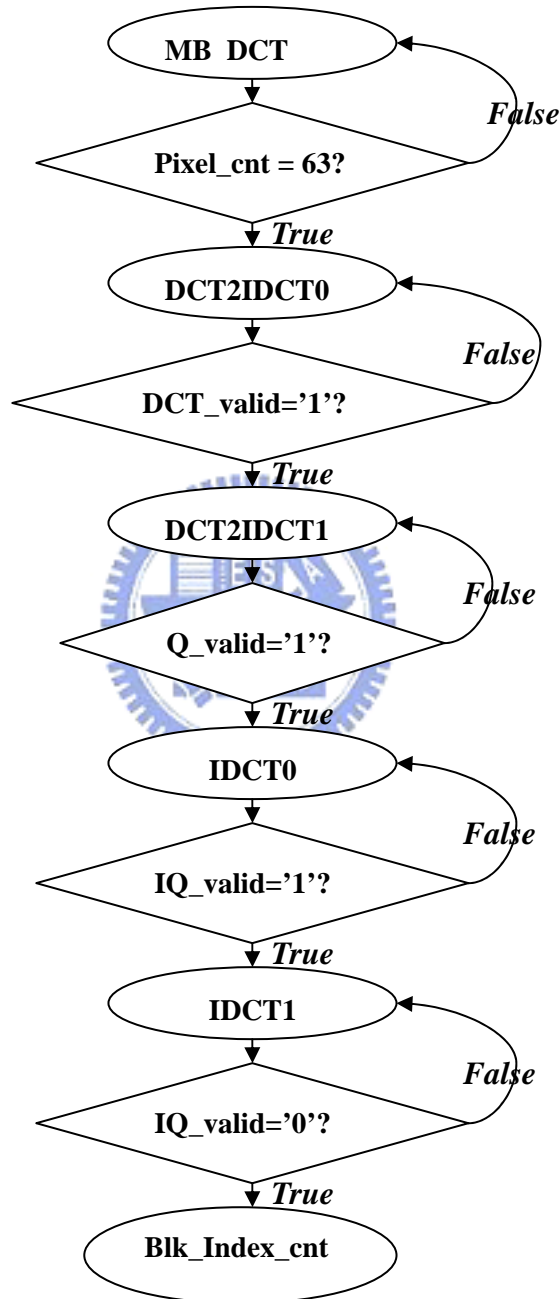


Figure 3-22 Block-level Operation Procedure

Chapter 4 ASIC Implementation

4.1 Design Flow

The traditional hardware design flow is depicted in Fig. 4-1. In the first instance, the module specification is determined and the computational complexity of each algorithm are investigated and analyzed by the C model. The C model adopted for our architecture is the XviD [19] MPEG-4 video encoder/decoder, which is a free software developed by the XviD organization. The designers can modify or redistribute the XviD MPEG-4 software in accordance with their requirements respectively. Some processes in C model are rewritten in hardware-like format consisting bit-width precision and data flow. It helps us to verify the hardware designs and find out the bugs. For instance, both the DCT and the IDCT have the same precision in the software and hardware model. The hardware architectures are implemented by using Hardware Description Language such as Verilog or VHDL. HDL simulation procedure is used to confirm that the results meet our requirement. The waveform simulation using Modelsim can analyze the timing and the signal values to correct the errors in the hardware model. When the function of the hardware model is correct, we can use Synopsys to synthesize these gate-level HDL codes. Then, the post-synthesis simulation is provided to check the timing specification. If the timing is not satisfied, the more tight constraints may be used or the hardware architecture may be modified and verified again. Before achieving our specification, these steps are repeated.

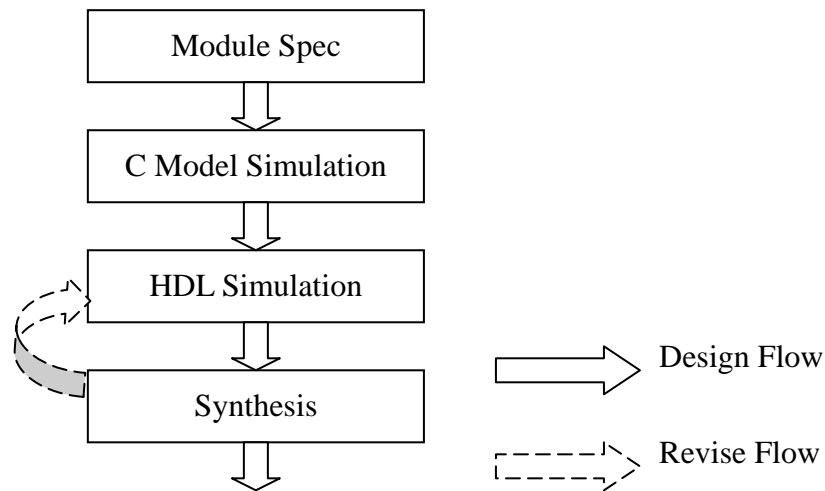


Figure 4-1 Module design flow

4.2 Functional Verification

Generally speaking, the design and the simulation in the software model are faster than that in the hardware model. Because the original DCT and IDCT algorithms in the XviD MPEG-4 video codec are not suitable for the hardware implementation, the C models of DCT and IDCT are required to be developed. When the C models of DCT and IDCT are implemented to emulate our DCT/IDCT hardware architecture, it makes the testing and verification of our proposed architecture more efficient. By decoding the encoded files, it is very easy to verify whether the C models of DCT and IDCT is correct or not. After the verification of the C models completely, the flowchart as shown in Fig. 4-2 is adopted to verify the architecture of the MPEG-4 texture coding. The testing patterns for functional verification of each module are obtained from the MPEG-4 C model. The hardware model is simulated with Modelsim developed by Mentor Graphics. Some information such as the waveform and the signal values can be displayed on the screen during the simulation. Dumping the information of the I/O signals to the text files will be efficient to check the errors. The output values of each module in software model and hardware model are written into text files. These files are manually

compared by using UltraEditor. If it encounters any mismatch in the hardware model and the software model, the HDL codes will be modified and then simulated again. The iteration continues until the comparison of the text file from hardware model and that from the software model is equivalent.

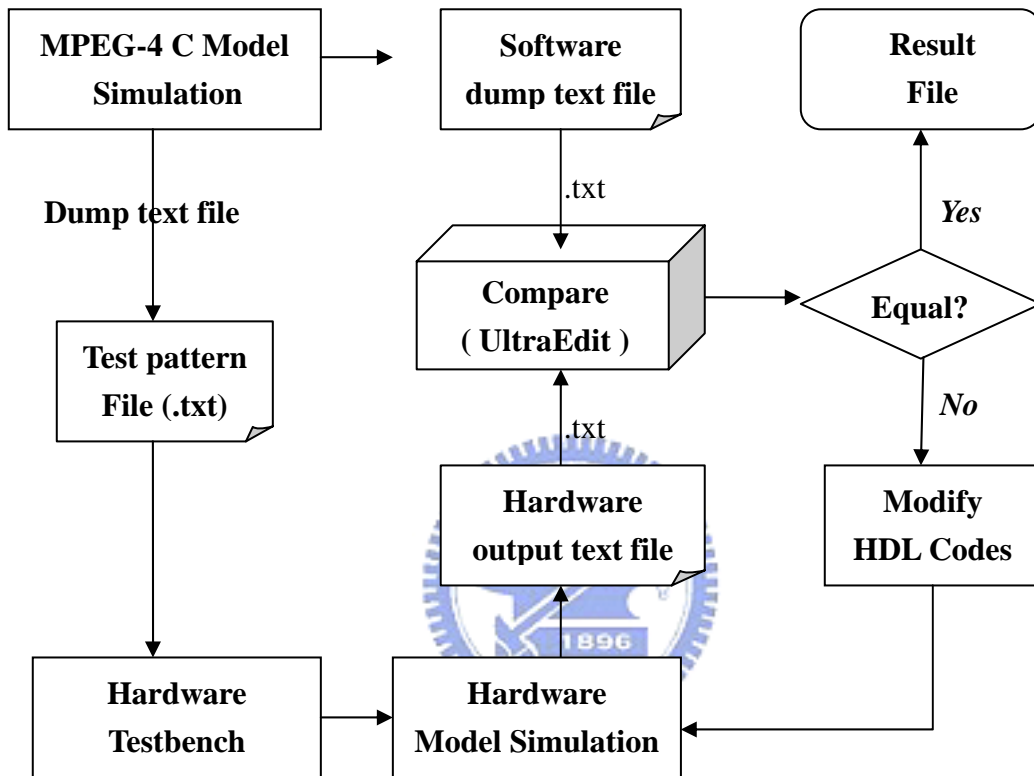


Figure 4-2 Flow chart of functional verification

◆ Design For Testability

After the logic synthesis of the MPEG-4 texture coding module is complete, design-for-testability is required to make an IC be testable. It involves inserting or modifying logic, and adding pins. The design-for-testability technique will reduce field returns, the complexity of test generation, the cost of testing, and improve yield. We can use the DFT compiler to achieve the design-for-testability. An overview of the DFT compiler flow is shown in Fig. 4-3. First, your scan style supported by vender's library must be selected. The scan style will tell the DFTC what type of scan-equivalent flip-flop is used in synthesizing the

logic. Second, Pre-scan DRC is to check gate-level scan design rule before the scan chain synthesis. We must fix the DFT violations if necessary. Third, scan-chain is inserted. Fourth, post-scan DRC is required to confirm that there are no new DFT problems. Besides, it can verify the scan chains synthesized operates properly, and create an ATPG-ready database. Finally, the TetraMAX is used to estimate Fault Coverage. The fault coverage of our design with DFT consideration can achieve 95.80%.

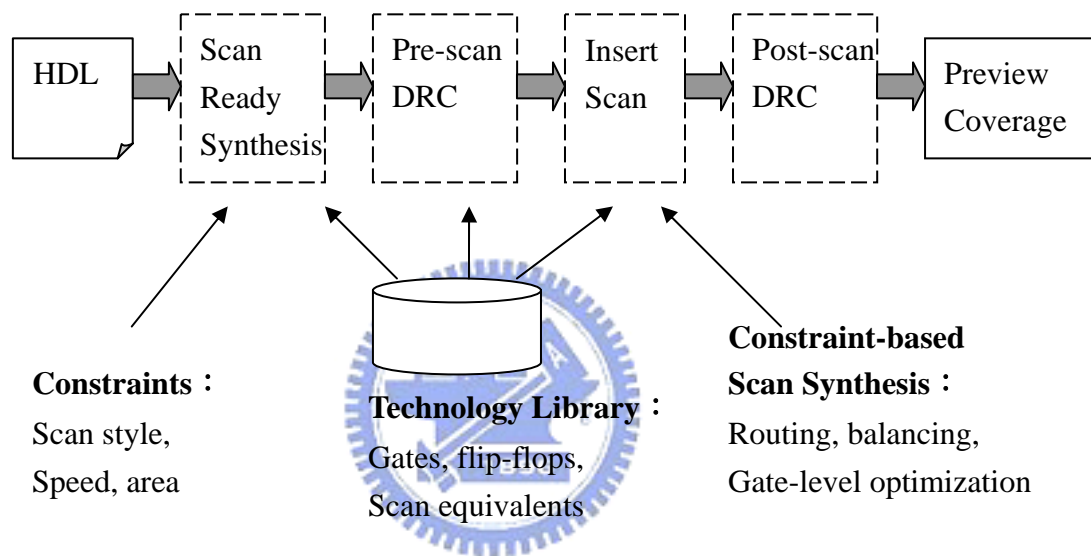


Figure 4-3 Overview of DFT compiler flow

Because the memory is regarded as a black-box model and unobservable for testing, the shadow wrapper is required to insert around the memory. SynTest SRAMBIST is adopted to support the memory testing. The memory BIST architecture is depicted in Fig. 4-4. The BistMode signal chooses testing or working mode in our design. When testing mode is provided, the data of the memory are obtained from the BIST controller. Since there are six memory modules in the MPEG-4 texture coding architecture, both the BistFail and ErrorMap have six bits to express all memories respectively. If any error occurs, it will be easily detected from these signals. When the chip is in the working configuration, these memories are regularly used to access data.

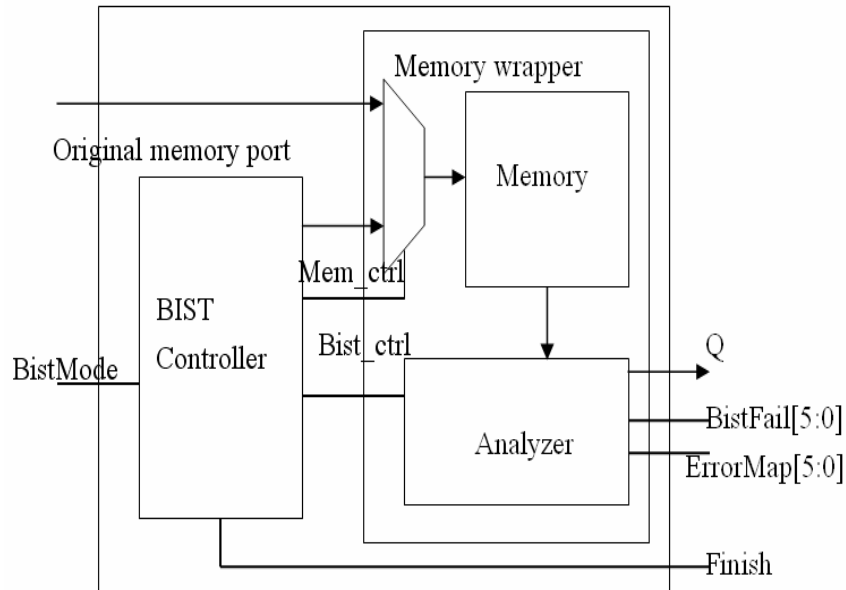


Figure 4-4 Memory Built-In Self-Test Architecture

◆ **IEEE IDCT accuracy measurement**

Fig. 4-5 shows the setup for measuring the accuracy of a proposed 8x8 IDCT. IEEE Std 1180-1990 specifies the numerical characteristic of 8x8 IDCT for visual telephony and similar applications where the 8x8 IDCT results are used in the reconstructed loop. For each 8x8 block, round the 64 resulting transformed coefficients to the nearest integer values and clip them to the range -2048 to 2047. For each of the output pixels of the 8x8 IDCT and for each of data sets of the 10,000 block generated for the definition, measuring the peak, mean, and mean square errors between the “reference” data and the “test” data. The random data from range (-300, 300), (-255, 256), (-5, 5) is input and the rates are generated in 5 items which are peak error (PE), mean square error (MSE), overall mean square error (OMSE), mean error (ME), and overall mean error (OME). Table 4-1 shows the results.

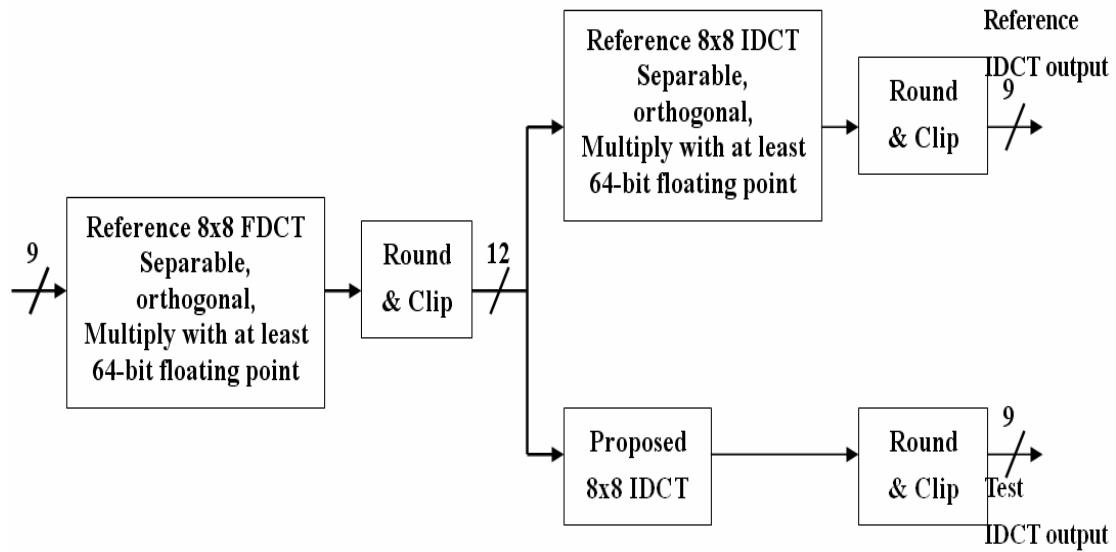


Figure 4-5 Setup for measuring the accuracy of a proposed 8x8 IDCT

Table 4-1 Accuracy Test result of the IDCT

Test Parameter	H:300 L:-300	H:255 L:-256	H : 5 L : -5	Sign invert 300	Sign invert 255	Sign invert 5	L=H=0	IDCT Spec.
OMSE	0.0145	0.0151	0.0095	0.0142	0.0153	0.0095	0.0000	<0.02
OME	0.0002	0.0001	0.0002	0.0002	0.0004	0.0002	0.0000	<0.0015
PE	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	= 1
MSE	0.0175	0.0180	0.0130	0.0167	0.0180	0.0127	0.0000	<0.06
ME	0.0074	0.0094	0.0090	0.0083	0.0086	0.0093	0.0000	<0.015

4.3 Implementation Result

The timing of the proposed architecture is shown in Fig. 4-6. The DCT/IDCT architecture has 97 cycles latency to get the output coefficients. After DCT, there are four pipeline stages in the quantizer unit and four pipeline stages in the inverse quantizer unit. Furthermore, one clock for status change in the texture coding FSM is required. The actual number required for a macroblock is 1137 cycles.

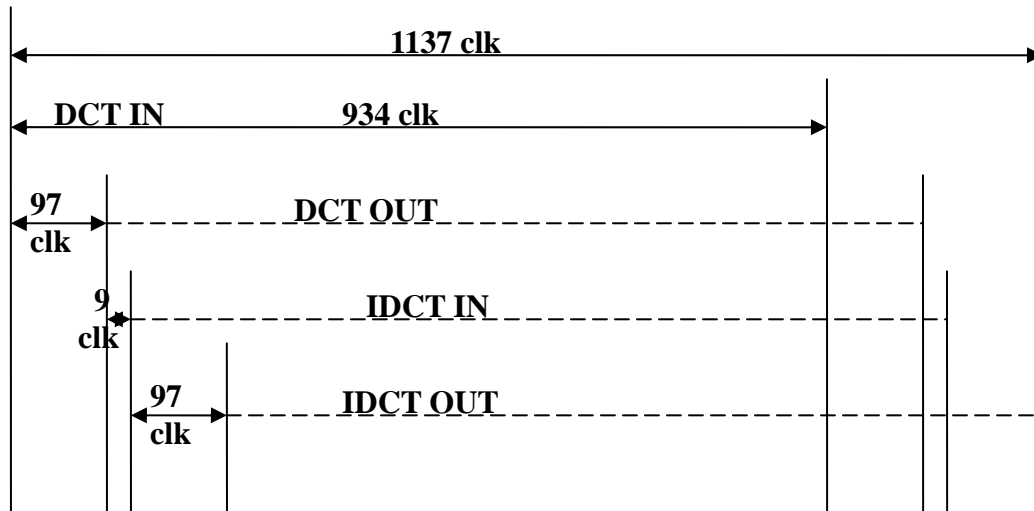


Figure 4-6 Timing of the proposed architecture

Table 4-2 depicts the implementation results for our texture coding engine. Due to the interleave DCT/IDCT scheduler, we adopt the sharing technique for AC/DC prediction with the Quantizer to reduce hardware cost. The area size of the DCT/IDCT architecture is larger because the multiplier-adder based architecture is adopted to implement this unit. It improves the working frequency but needs more area cost. Only one 64x16 transpose memory in the DCT/IDCT architecture is required in our work. The total ram size of our proposed is less than K. Suh's. When the latency of the SDRAM are 5 clock cycles and the working frequency is 21 MHz, our texture coding engine meets the real-time requirement for encoding the frame sequences of CIF (352x288) at 30 fps with 54,405 gates. Furthermore, the maximum working frequency in our design is 43 MHz.

Table 4-2 Comparison with previous work

Our Work			K. Suh's		
1137 cycles for one macroblock and the working frequency is 21 MHz			1064 cycles for one macroblock and the working frequency is 27 MHz		
Module	Logic (Gate)	RAM (bit)	Module	Logic (Gate)	RAM (bit)
DCT/IDCT	27,061	64x16	FDCT	7,005	64x16
			IDCT	8,091	64x16
Q	4,132	0	Q/IQ	3,514	0
IQ	2,323	0			
AC/DC prediction	11,304	742x12	AC/DC Prediction	17,939	742x12
Control	7,980	0	AMBA interface	2,790	0
Scan logic	1,605	0	Scan logic	2,841	0
Q. coeff buffer	0	32x16x3 +32x8x4	Q. coeff buffer	0	384x12
Total	54,405	12,488	Total	42,180	15,536

Table 4-3 lists the SRAMs required for each module. In the AC/DC prediction module, there is a 742x12 bits SRAM for storing the prediction values. In the ping-pong buffer, two SRAM are used for buffering the current macroblock data and the next macroblock data. 9 bits are necessary for expressing each pixel. There is one transpose memory in the DCT/IDCT module, and its characteristic is dual-port. Dual port SRAM has two read/write ports, and can be read and written simultaneously. Total SRAM used in the texture coding unit is 16,840 bits.

Table 4-3 Memory required for each module

Functions	Characteristic	Depth x Width	Num.	Bits
ACDC Prediction		742x12	1	8,904
Transpose mem. for DCT/IDCT	Dual port	64x16	1	1,024
Ping-pong buffer		96x36	2	6,912
Total			4	16,840

The synthesized gate count of each module is shown in Fig. 4-7. Total gate count is 54,405 gates. The DCT/IDCT module has 27,061 gates and occupies about half logic gates in the texture coding system. The AC/DC prediction without multiplier has 11,304 gates. This module needs some registers to store the prediction or quantized (non- prediction) data. The controller in the top module is designed to control each module and requires 7,980 gates. The quantizer and inverse quantizer which both are implemented with the multipliers have 4,132 gates and 2,323 gates, respectively.

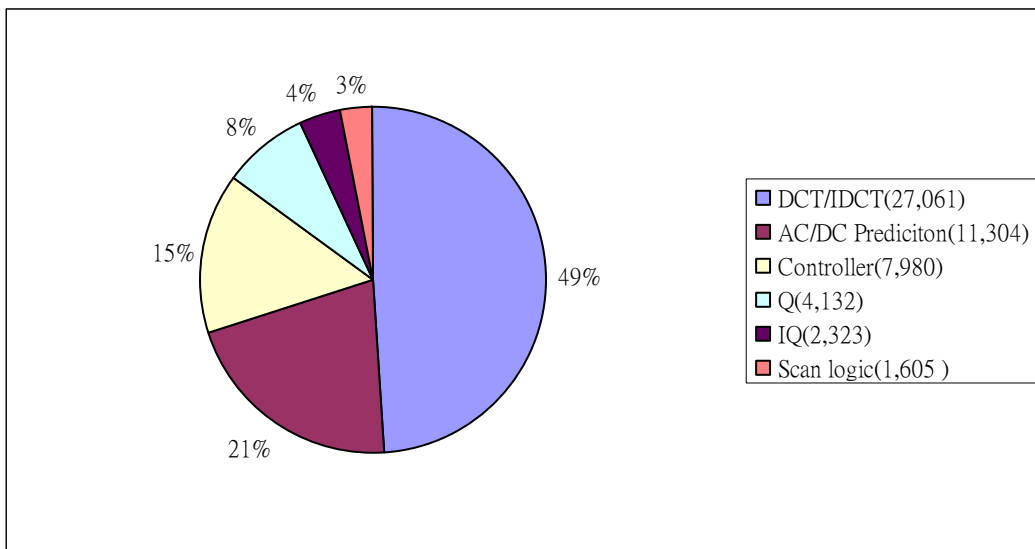


Figure 4-7 Gate Count of each module

When replacing the XviD C model by our texture coding module, the subject view is necessary to guarantee that the quality of the reconstructed frames is excellent. Fig. 4-8 to Fig. 4-11 show the reconstructed frames of the testing sequence with our texture coding engine. The testing environment is under the condition of 384K bit-rate, 30 frames per second, and 300 frames between each interval key-frame. The IDCT precision satisfies IEEE IDCT precision specification and the average PSNR of our reconstructed frames are very close to that of the XviD version. The average PSNR during the encoding using our texture coding

engine compared to that of the XviD version with four testing sequence are shown in Table 4-4. As we can see, there is only little PSNR degradation in our work. It is still indistinguishable between these two for human vision.

Table 4-4 the average PSNR of our work and that of the XviD version

	Our work	XviD
Akiyo CIF	41.43782(-0.06508)	41.5029
Foreman CIF	29.10254(-0.00819)	29.11073
Table CIF	29.72092(-0.01889)	29.73981
News CIF	33.85705(-0.04639)	33.90344



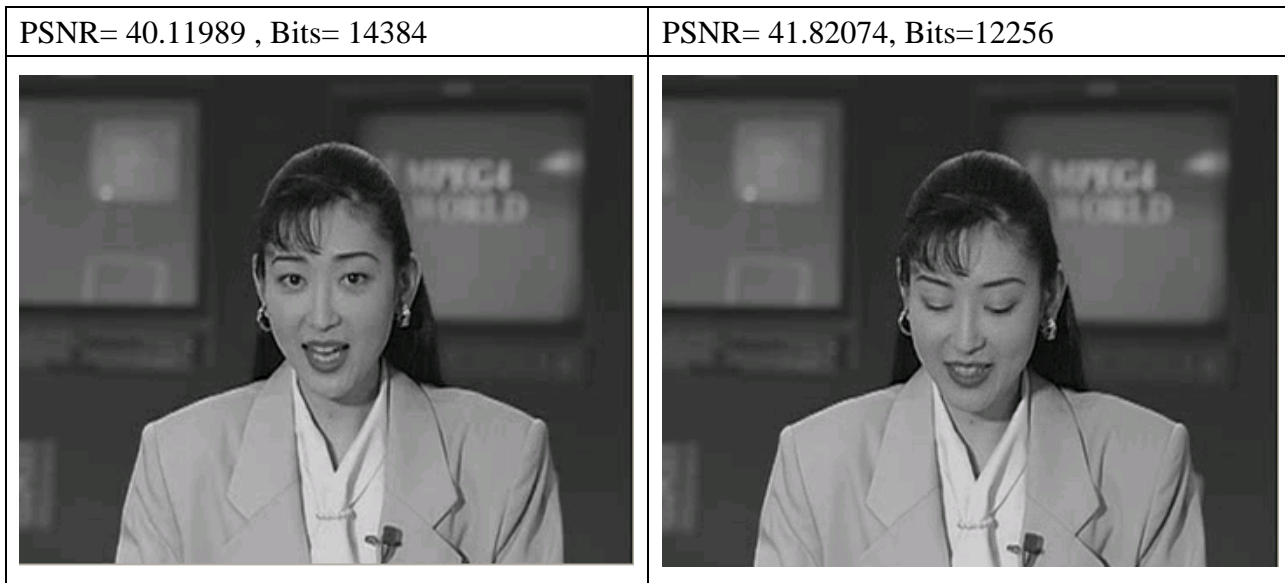


Figure 4-8 Subject view of reconstructed frame for akiyo sequences at the 46th frame and 241st frame

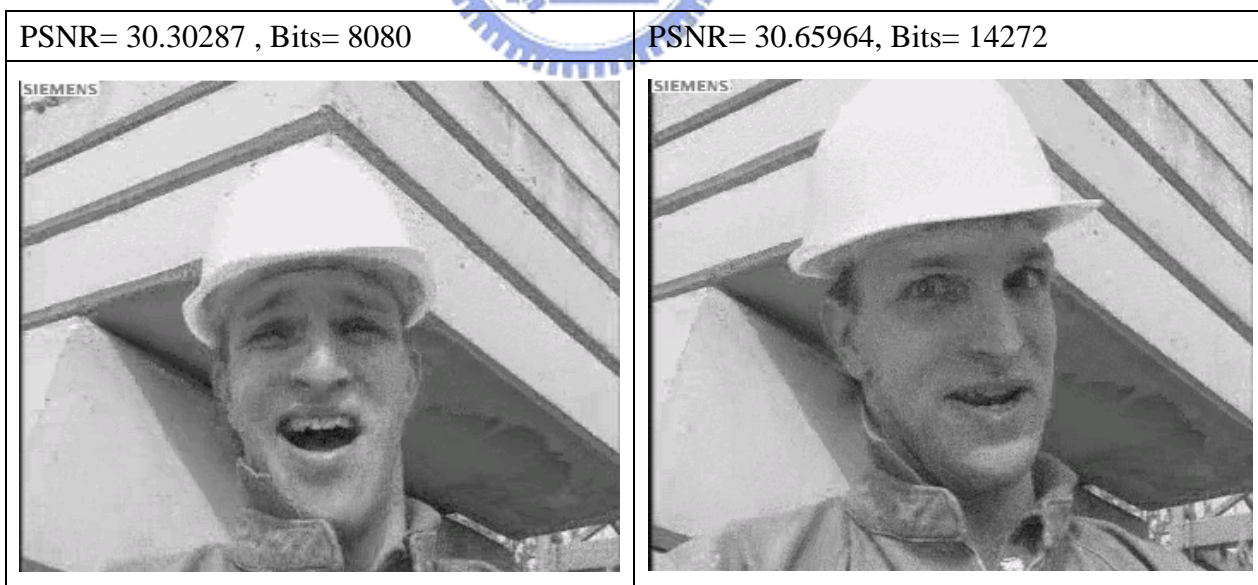


Figure 4-9 Subject view of reconstructed frame for foreman sequences at the 27th frame and 109th frame

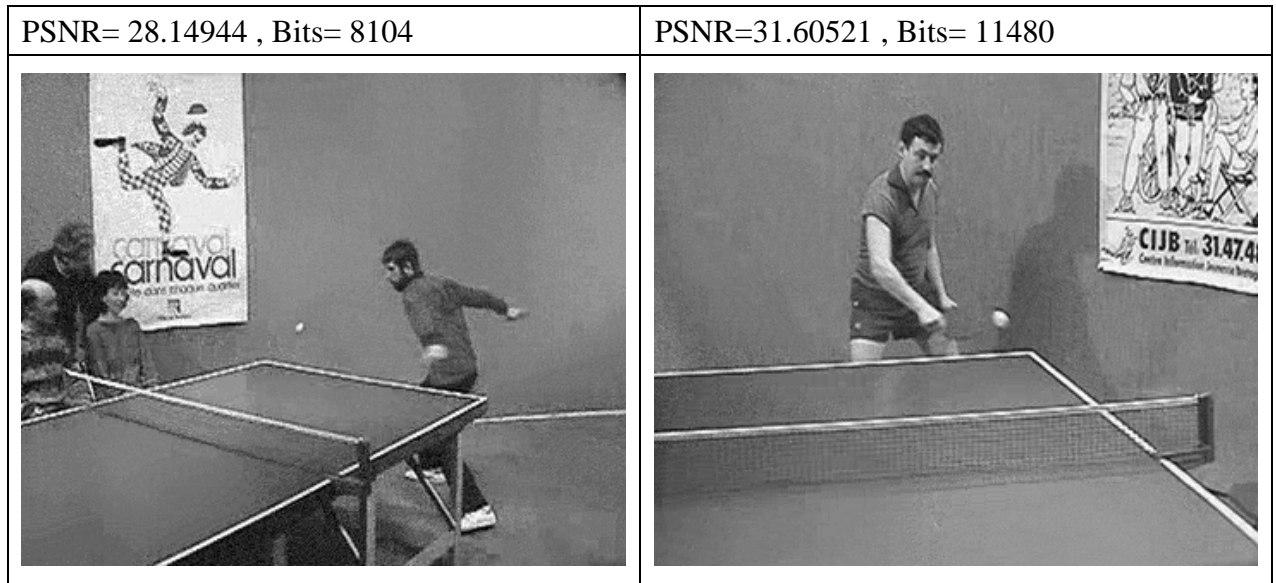


Figure 4-10 Subject view of reconstructed frame for Table sequences at the 122nd frame and 183rd frame



Figure 4-11 Subject view of reconstructed frame for News sequences at the 21st frame and 146th frame

Chapter 5 Conclusion

In this thesis, the hardware architecture for the texture coding module in the MPEG-4 video encoder is presented. This proposed hardware core can support Simple Profile Level 3, under frame size 352x288 with 30fps for real time video applications. In order to reduce the hardware cost and smaller the processing time, an efficient block engine using the interleaving DCT/ IDCT scheduling is adopted. While this module is integrated into the entire system, it will maintain performance in low cost. Furthermore, the sharing technique for normalizing the AC/DC prediction values and quantizing DCT coefficients is applied to reduce area size further. The ping-pong buffer is designed to buffer motion estimation errors or intra frame data and ensure that all the data in the buffer can be read safely and correctly. Based on the row-column decomposition technique, the cost-effective VLSI architecture for two-dimensional 2D DCT/ IDCT is achieved. The 2D DCT/IDCT design has a regular structure, simple interconnects and control, and efficient implementation of the inverse transform using the same hardware. The required finite word-length accuracy is analyzed. The DCT/IDCT structure can achieve excellent accuracy and the accuracy conforms to IEEE standard 1180- 1990.

In summary, a cost-effective block engine for MPEG-4 texture coding is presented and achieved. As the architecture can be placed in the regular fashion, it is proper to be implemented with commercial ASIC technologies. The proposed architecture can be applied to the portable multimedia terminal for wireless multimedia services. The future work includes three tasks. First, some improvements in our work to lower power consumption are necessary. These methods include clock gating, skipping input macro block for DCT or IDCT in the encoding loop...etc. Second, more functionality such as error-resilience tools, B-VOP coding will be integrated into the original module to satisfy other video applications. Third,

the decoding functions, such as variable length decoding, will also be designed to integrate into the original architecture to be a MPEG-4 texture codec design.



Reference

- [1] MPEG-4 Video Group "Information Technology - Coding of Audio Visual Object-Part2: Visual," *ISO/IEC JTC 1/SC 29/WG 11 M9477*, Pattaya, March 2003.
- [2] ITU-T Recommendation H.263, "Video coding for low bit rate communication," *ITU-T*, 1996
- [3] ISO/IEC JTC1 IS 11172, "Coding of Moving Picture and Coding of Continuous Audio for Digital Storage Media up to 1.5 Mbps," *ISO/IEC JTC1*, 1992.
- [4] ISO/IEC JTC1/SC29/WG Draft CD 13818-2, "General Coding of Moving Pictures and Associated Audio," *ITU-T Recommendation H.262 Committee Draft*, 1994.
- [5] Fernando Pereira and Touradj Ebrahimi, "The MPEG-4 Book," Upper Saddle River, NJ : Prentice Hall PTR, c2002.
- [6] T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.7, No.1, pp.19-31, Feb. 1997.
- [7] Atul Puri and Tsuhan Chen, "Multimedia systems, standards, and networks," Marcel Dekker, New York, c2000.
- [8] A.V. Oppenheim and R.W. Shafer, "Digital Signal Processing," Prentice Hall, N.J., Englewood Cliffs, 1975.
- [9] J. Canaris, "A VLSI Architecture for the Real Time Computation of Discrete Trigonometric Transforms," *Journal of VLSI Signal Processing*, vol. 5, pp. 95-104, 1993.
- [10] Weiping Li, "A New Algorithm to compute the DCT and its Inverse," *IEEE Trans. on Signal Processing*, Vol. 39, pp 1305-1313, June 1991.
- [11] M. Bousselmi et al., "New parallel architecture of the DCT and its inverse for image compression," *IEEE International Conference on Electronics, Circuits and Systems (ICECS 2000)*, vol. 1, pp. 345-348, Dec. 2000.
- [12] A. Madisetti, A. N. Willson Jr., "A 100 MHz 2-D 8x8 DCT/IDCT Processor for HDTV

Applications," *IEEE Trans. On Circuits and Systems for Video Technology*, vol.5, No.2, pp. 158-165, Apr. 1995.

- [13] K. H. cheng et al., "The Design and Implementation of DCT/IDCT Chip with Novel Architecture," *IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, Geneva Switzerland, vol. 4, pp. 741-744, May 28-31, 2000.
- [14] C.W. Hsu, W.M. Chao, Y.C. Chang, and L.G. Chen, "Texture coder design of MPEG-4 video by using interleaving schedule," in *Proc. of 2002 IEEE International Conference on Multimedia and Expo (ICME 2002)*, Lausanne, Switzerland, August 2002.
- [15] Seehyun Kim, Wonyong Sung "Fixed-Point Error Analysis and Word Length Optimization of 8x8 IDCT Architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, No. 8, Dec. 1998.
- [16] E. E. Swartzlander, Jr., "Truncated multiplication with approximate Rounding," in *Conference Record of the Thirty-third Asilomar Conference on Signals, Circuits and Systems*, pp.1480-1483, 1999.
- [17] IEEE standard 1180-1990 "IEEE Standard Specifications for the Implementation of 8x8 Inverse Discrete Cosine Transform," *CAS Standards Committee of the IEEE Circuits and Systems Society*, Dec. 6, 1990.
- [18] K. Suh, S. Park, S. Kim, B. Koo, I. Kim, K. Kim, H. Cho, "An Efficient Architecture of DCTQ Module in MPEG-4 Video Codec," *IEEE International Symposium on Circuits and Systems(ISCAS 2002)* , vol. 1, pp. I-777 – I-780, 2002.
- [19] <http://www.xvid.org>

Appendix

A-1 . Pin Definitions for MPEG-4 texture coding

Fig. A-1 shows the inputs/outputs of our MPEG-4 texture coding. The descriptions for each pin are depicted in Table A-1.

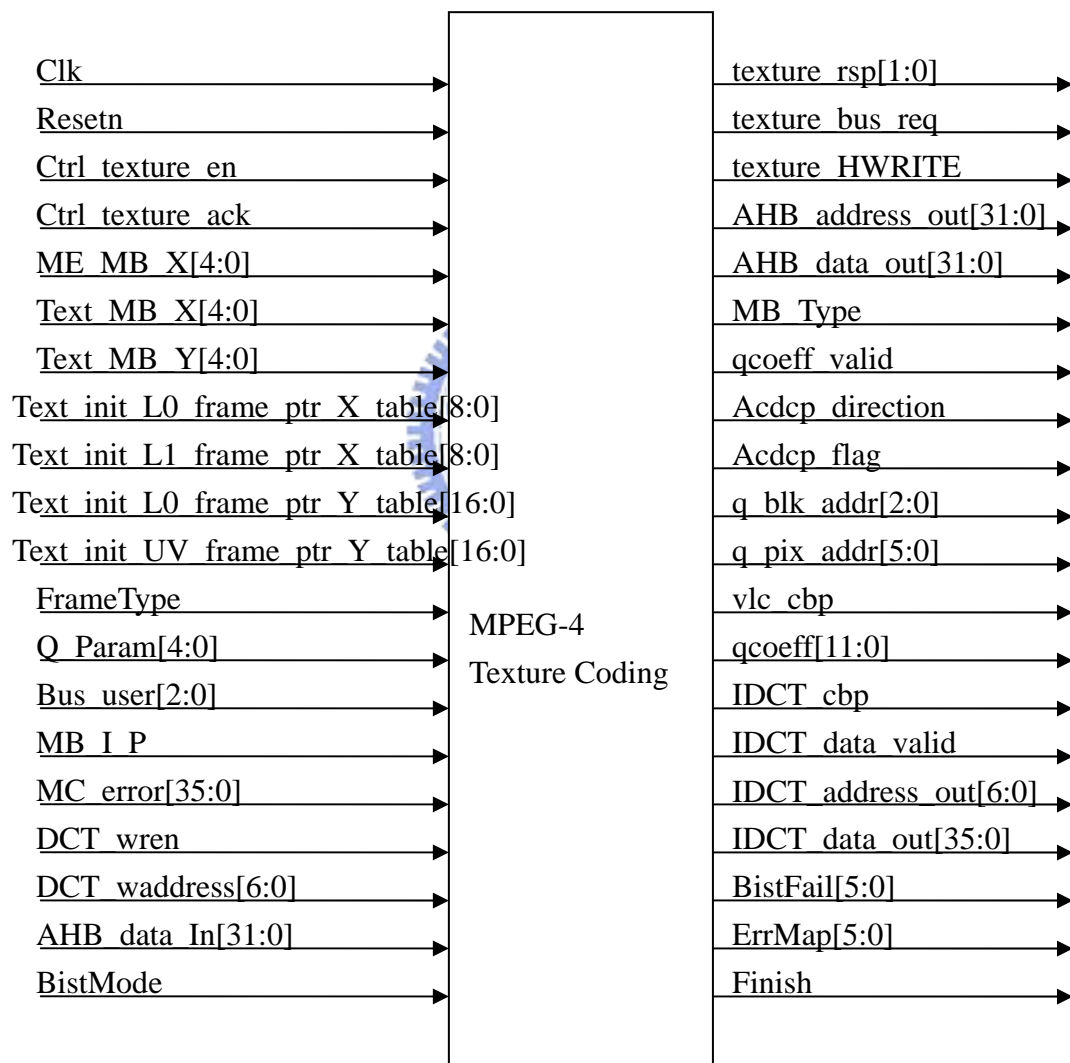


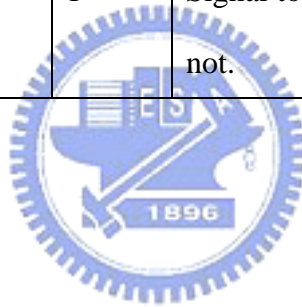
Figure A-1 MPEG-4 Texture Coding IP.

Table A-1 PIN Definition

Name	Direction	Width	Description
Clk	Input	1	Clock signal, positive edge trigger
Resetn	Input	1	Reset texture coding engine, active LOW
Ctrl_texture_en	Input	1	Enable Texture coding engine
Ctrl_texture_ack	Input	1	Acknowledge signal
ME_MB_X	Input	5	the MB position in the X-axis for Motion estimation
Text_MB_X	Input	5	the MB position in the X-axis for Texture coding
Text_MB_Y	Input	5	the MB position in the Y-axis for Texture coding
Text_init_L0_frame_ptr_X_table	Input	9	Offset of the frame pointers for the luminance values in the X-axis
Text_init_L1_frame_ptr_X_table	Input	9	Offset of the frame pointers for the chrominance values in the X-axis
Text_init_L0_frame_ptr_Y_table	Input	17	Offset of the frame pointers for the luminance values in the Y-axis
Text_init_UV_frame_ptr_Y_table	Input	17	Offset of the frame pointers for the chrominance values in the Y-axis
FrameType	Input	1	Frame type I or P
Q_param	Input	5	Quantization Parameter
bus_user	Input	3	AMBA bus user
MB_I_P	Input	1	MB type intra or inter
MC_error	Input	36	Motion compensation errors, store MC

			errors in the ping-pong buffer if current frame is the Prediction frame.
DCT_wren	Input	1	Enable to write MC errors to ping-pong buffer
DCT_wraddress	Input	7	Address for writing MC errors to ping-pong buffer, address [6: 4] indicates the block index and address [3:0] indicates the pixel index.
texture_rsp	Output	2	Response to tell the status of texture coding engine(idle, busy, or finish)
texture_bus_req	Output	1	Request to use the AMBA bus
AHB_data_in	Input	32	Input data from the frame memory through the AMBA bus
texture_HWRITE	Output	1	Writing the reconstructed values to the frame memory through the AMBA bus
AHB_address_out	Output	32	Address of the reconstructed frame
AHB_data_out	Output	32	Reconstructed frame data for I frame
MB_type	Output	1	MB type to VLC module
qcoeff_valid	Output	1	Valid signal for quantized coefficients to VLC module
acdcp_direction	Output	1	Prediction direction
acdcp_flag	Output	1	Prediction flag to decide whether the prediction values are used or not.
q_blk_addr	Output	3	Quantization block address bus
q_pix_addr	Output	6	Quantization pixel address bus

vlc_cbp	Output	1	Cbp signal to VLC
qcoeff	Output	12	Quantized coefficients to VLC
IDCT_cbp	Output	1	Cbp to Motion compensation
IDCT_data_valid	Output	1	Valid signal to the reconstructed memory in the Prediction frames
IDCT_address_out	Output	7	the MC errors Address bus
IDCT_data_out	Output	36	the reconstructed frame data bus
BistFail	Output	6	BIST fail
ErrMap	Output	6	Error mapping for BIST mode
Finish	Output	1	BIST finish signal
BistMode	Input	1	Signal to determine if BIST mode is used or not.



Biography

Yao-Chun Hung was born in Changhua, Taiwan, R.O.C., on March 25, 1981. He received the B.S and M.S degrees from the Department of Electrical and Control Engineering, National Chiao Tung University, Taiwan, in 2003 and 2005 separately. His research interests include video coding algorithms and VLSI architecture for image and video processing.



Publication List

- [1] Bing-Fei Wu, **Yao-Chun Hung**, Yen-Lin Chen, Chao-Jung Chen, Chung-Cheng Chiu, and Chorng-Yann Su, "A High-Speed Wavelet-Based Video Codec for Video Surveillance System", 2004
第十三屆全國自動化科技研討會, Taipei, Taiwan, June 17~18



Awards

- [1] 交通大學電機與控制工程學系 88 學年度第二學期書卷獎
- [2] 第五屆 TIC 100 創業競賽 冬令營 冠軍
- [3] 第五屆 TIC 100 創業競賽 總決賽 銀質獎
- [4] 教育部九十二學年度大專院校通訊科技競賽研究所組入圍
- [5] 第一屆機動車輛創新設計獎(智慧電子化機能創新設計組) 銀質獎