

國立交通大學

電機學院 IC 設計產業研發碩士班

碩士論文

多重影像標準應用之反向離散餘弦轉換設計



Design of an Inverse Discrete Cosine Transform Core for
Multiple Video Standards Applications

研究生：洪岳琪

指導教授：李鎮宜 教授

中華民國九十六年元月

多重影像標準應用之反向離散餘弦轉換設計

Design of an Inverse Discrete Cosine Transform Core for
Multiple Video Standards Applications

研究生：洪岳琪

Student：Yueh-Chi Hung

指導教授：李鎮宜 教授

Advisor：Prof. Chen-Yi Lee

國立交通大學

電機學院 IC 設計產業研發碩士班



Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on
IC Design

January 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年元月

多重影像標準應用之反向離散餘弦轉換設計

研究生：洪岳琪

指導教授：李鎮宜 教授

國立交通大學

電機學院產業研發碩士班



在本論文中，提出一組硬體共用之反向離散餘弦轉換的架構，包含了三種不同影像壓縮的標準：MPEG-2 與 H.264/AVC High Profile 與 H.264/AVC Baseline。並且提出一套演算法來計算反向離散餘弦轉換的矩陣，以及資料流的配置，我們的提案由 MPEG-2 為基礎，來推導出 H.264/AVC 的資料流配置，並且以心臟收縮陣列的方法及加法器與移位器計算矩陣乘法，並且提出一套演算法化簡乘法器的數量，已達到減少硬體面積的目標，我們所提出來的架構基於 0.18 微米聯華電子 (UMC) 互補式金氧半導體製程，我們的硬體共用之反向離散餘弦轉換的硬體設計需要包含 72800 個邏輯閘，可運作在 100MHz 的時脈以上。

Design of an Inverse Discrete Cosine Transform Core for Multiple Video Standards Applications

Student : Yueh-Chi Hung

Advisor : Prof. Chen-Yi Lee

**Industrial Technology R & D Master Program of
Electrical and Computer Engineering College**

National Chiao Tung University



ABSTRACT

In this thesis, we propose a hardware-shared architecture for inverse discrete cosine transform (IDCT). It includes three different video compression standards: MPEG-2, H.264/AVC baseline and H.264/AVC high profile. Then offer an algorithm to calculate IDCT matrix and assign the data flow. Our proposal based on MPEG-2 to decide the data flow of H.264/AVC. Uses one dimension systolic array method, additions and subtractions to calculate matrix multiplication to perform the hardware area reduction.

The proposed architecture based on 0.18 μ m UMC CMOS Process, our IDCT design needs 72800 gate counts and operates over 100MHz.

誌

謝

首先感謝恩師 李鎮宜教授在交通大學研究所求學兩年的細心指導，建立且加深了積體電路設計需要由系統角度設計的觀念，在每次的會議報告及討論中，給予寶貴且實際的意見，使得研究能順利的完成，並且在Si2 (System Integration and Silicon Implementation Group；<http://si2lab.org/>)實驗室中提供了充裕的積體電路設計環境及資源，可以安心無餘地從事研究工作。

再來是感謝交通大學機械系 蔡忠杓教授的協助與引薦，以及康正、毅宏、俊彥、皆賢、秉昌、堉棋、大嘉、名威、文平、韋磬、昱德，及其他 Si2 實驗室學長、同學們的指導幫助，彥欽、思慧、紹航及華鼎等 IC 設計產業研發碩士班同學的相互鼓勵、產業研發碩士班助理 佩瑾、義隆電子 魏誠文主任及摯友 盈成的協助。

最後更要感謝我的 父母、岳父母、妹妹 一菁的關心與協助，尤其是摯愛的妻子 惠雯全力的支持，讓我在婚後能無後顧之憂地一圓唸書的夢想。

謹將此論文獻給關心及愛護我的家人與朋友們。

Contents

Chapter 1	Introduction.....	1
1.1	Motivation.....	1
1.2	Organization of this thesis.....	2
Chapter 2	Overview of Inverse Discrete Cosine Transform (IDCT).....	3
2.1	IDCT in Decoder for Different Standards	3
2.2	Overview of MPEG Compression Algorithm	4
2.2.1	Temporal Redundancy Reduction	4
2.2.2	Spatial Redundancy Reduction.....	5
2.2.3	The Process of Decoding for MPEG-2.....	6
2.3	Algorithms of Inverse Discrete Cosine Transform.....	7
2.3.1	Direct Computation of Two Dimension DCT	8
2.3.2	Parallel Implementations	9
2.4	Paper Reference.....	13
2.5	The Proposed Algorithm.....	16
2.5.1	IDCT of MPEG-2	18
2.5.2	IDCT of H.264 High Profile	20
2.5.3	IDCT of H.264 Baseline	22

Chapter 3	Architecture Overview.....	23
3.1	Propose System Architecture of IDCT	23
3.2	Overview of One Dimension Systolic Array	26
3.3	Refinement of Matrix Multiplication	31
3.3.1	Canonical Signed Digit (CSD) and Modified Canonical Signed Digit (Modified CSD)	31
3.3.2	Zero Skip and DC Value Skip	35
Chapter 4	Simulation and Implementation Results.....	38
4.1	Simulation Results.....	38
4.2	Implementation Results.....	41
Chapter 5	Conclusions	46
Bibliography	47
About the Author	49

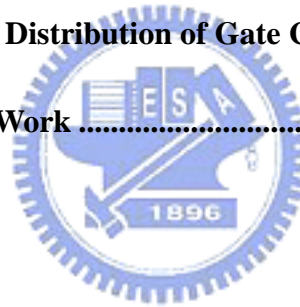


List of Figures

Fig.2.1	IDCT in MPEG-2 Decoder	3
Fig.2.2	IDCT in H.264 Decoder.....	4
Fig.2.3	The Temporal Picture Structure.....	5
Fig.2.4	MPEG-2 Decoding Process	6
Fig.2.5	Zig-zag Scan Order	7
Fig.2.6	Signal Flow Graph of Direct Computation Algorithm for N=8.....	9
Fig.2.7	The Example of Bayoumi's [10] Architecture for 5 Point DFT.....	10
Fig.2.8	The Function of Each PE	10
Fig.2.9	The Modified Systolic Architecture for 5 Point DFT	12
Fig.2.10	Systolic Architecture for 5 Point DFT	12
Fig.2.11	Processing Element for IDCT Even and Odd Matrix from D.W Kim [14].....	14
Fig.2.12	A. Madisetti's [15] Process Architecture	14
Fig.2.13	Data Recorder Unit (DRU) from [15].....	15
Fig.2.14	ACF Matrix-Vector Multiply Unit from [15].....	15
Fig.2.15	BDEG Matrix-Vector Multiply Unit from [15].....	15
Fig.2.16	J.I Guo's [16] Architecture	16
Fig.2.17	The Flow Graph of FCT for N=8	18
Fig.2.18	The 8×8 Matrix of MPEG-2 IDCT.....	19

Fig.2.19	The Transport for 8×8 Matrix.....	19
Fig.2.20	Separate 8×8 MPEG-2 IDCT into 4×4 Matrix.....	20
Fig.2.21	The Original 8×8 Matrix of H.264 High Profile IDCT	21
Fig.2.22	The Modified Column #5~#8 in Multiplicand Matrix and Row #5~#8 in Multiplier Ones of H.264 High Profile IDCT	21
Fig.2.23	Separate 8×8 H.264 High Profile IDCT into 4×4 Matrix.....	21
Fig.2.24	The 4×4 Matrix of H.264 Baseline IDCT.....	22
Fig.3.1	IDCT Operating Procedure	24
Fig.3.2	IDCT Block Input Interface	24
Fig.3.3	Block Diagram of Prototype IDCT Architecture.....	25
Fig.3.4	One-D Systolic Array Data Flow for Motion Estimation [11]	26
Fig.3.5	One-D Systolic Array Data Flow for 4×4 Matrix Multiplication	27
Fig.3.6	One-D Systolic Array Architecture for Matrix Multiplication.....	28
Fig.3.7	One-D Systolic Array Architecture Refinement for Matrix Multiplication	29
Fig.3.8	The Operation Cycles and Latency of Two-D 4×4 Matrix added more PE's	30
Fig.3.9	The Operation Cycles and Latency of Two-D 8×8 Matrix added more PE's	30
Fig.3.10	The CSD Conversion Algorithm	32
Fig.3.11	New Version IDCT Architecture Using Replaced Multiplication.....	35
Fig.3.12	DC Value Skip	36

Fig.3.13	Zero Value Skip.....	36
Fig.3.14	Zero/DC Value Method	37
Fig.3.15	A Variable-Length FIFO for the Synchronization after IDCT Output Interface.....	37
Fig.4.1	PMSE Value	40
Fig.4.2	PME Value.....	40
Fig.4.3	OMSE Value.....	40
Fig.4.4	OME Value	41
Fig.4.5	The Bar Graph of Gate Counts Reduction in Different Methods....	42
Fig.4.6	The Percentage Distribution of Gate Counts for Each Part.....	43
Fig.4.7	Layout of This Work	45



List of Tables

Table 3.1	Estimation Worst Cycle Counts	28
Table 3.2	An Example of Results For The Standard CSD And Modified CSD Conversions [12]	33
Table 3.3	The Binary, CSD, and Modified CSD Values of MPEG-2 Standard	33
Table 3.4	The Binary, CSD, and Modified CSD Values of H.264 Standard.....	34
Table 3.5	Estimate Operators for 4×4 Matrix Multiplication Architecture	34
Table 4.1	Estimate Bit Amount to Describe Floating Point of MPEG-2 Coefficients	39
Table 4.2	Gate Counts Reduction in Matrix Calculation	42
Table 4.3	Gate Counts of Each Part and Comparison.....	43
Table 4.4	Operators List and Compare with State-of-the-Art Works.....	44
Table 4.5	Compare with Other Approaches	44

Chapter 1

Introduction

1.1 Motivation

Science and technology evolve with each passing day, and people enjoy the convenience brought about by the development in technology. Many electronic products, such as VCD players, DVD players, Digital Cameras, and Digital Video Camcorders have been widely used in our daily life. Nowadays, many mobile products even include video playback capability, many and different video coding standards have frequently been proposed. Scheme one architecture provides more different standards can reduce the cost and use in a wider range of different products.

The MPEG-2 standard was defined by Motion Pictures Experts Group (MPEG). It is the most popular video format used for recoding and transmitting video data. In one specific area, the MPEG technology is used for the encoding and decoding of motion color pictures on TV. The MPEG-2 technology extends the application spectrum to include broadband Integrated Services Digital Network (ISDN), Direct Broadcast Satellite (DBS), and is used today in different applications such as DVD and High Definition Television (HDTV).

H.264/AVC was developed by the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG), and it is the newest video coding

standard. It achieves high coding efficiency by employing a number of new technologies. The main benefits are the compression performance in existing applications and capability to offer quality video over the internet. In addition, this technology is used in High Definition Television (HDTV), DVD, and mobile products that include video playback functionality.

Different video coding standards have frequently been proposed. However, these different standards are mostly not compatible with each other, and new standards are not backward compatible with older products that use technologies of the past most of the time. Based on Inverse Discrete Cosine Transform (IDCT), if we find a set of rules that incorporate the different standards, especially those more popular (e.g. MPEG-2) and newer (e.g. H.264 high profile and baseline) ones, one hardware architecture can then be employed to include the different standards. In addition, flexibility in the multimedia chips to incorporate new standards in the future will reduce the cost while providing more functions. This chip can be designed more easily and used in a wider range of applications.



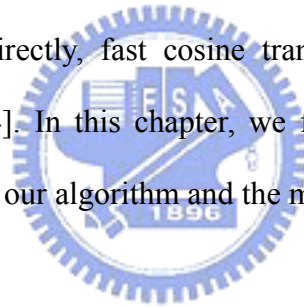
1.2 Organization of this thesis

This thesis is organized as follows. In Chapter 2, we present the IDCT algorithm. It contains the IDCT basics and the previous work. In addition, we also present the algorithm. Chapter 3 shows the proposed architecture of IDCT design, the matrix calculation algorithm, architecture and optimization. The verification method and simulation result will be shown in Chapter 4. We make a brief conclusion and future work in the last chapter.

Chapter 2

Overview of Inverse Discrete Cosine Transform (IDCT)

In 1974[1], DCT/IDCT is widely used in many video compression applications and standards; the goal of compression is to reduce redundancy. Many different algorithms have been proposed. Among the algorithms, they can be briefly as following: computing DCT directly, fast cosine transform, Memory-based designs [2~3], Adder-based designs [4]. In this chapter, we first introduce the definition of DCT/IDCT and other methods, our algorithm and the method for different matrix type.



2.1 IDCT in Decoder for Different Standards

The IDCT in the decoder for different standards, Fig.2.1 is the IDCT in MPEG-2 standards and Fig.2.2 is in H.264 standards.

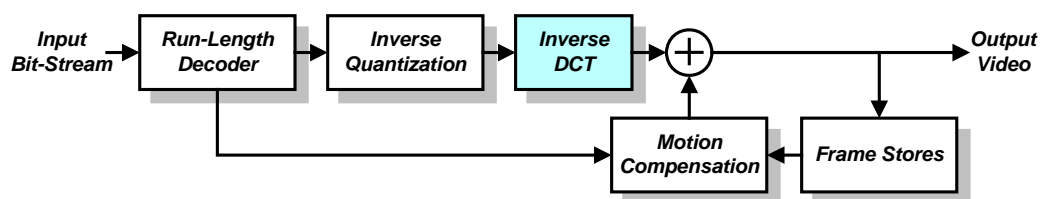


Fig.2.1 IDCT in MPEG-2 Decoder

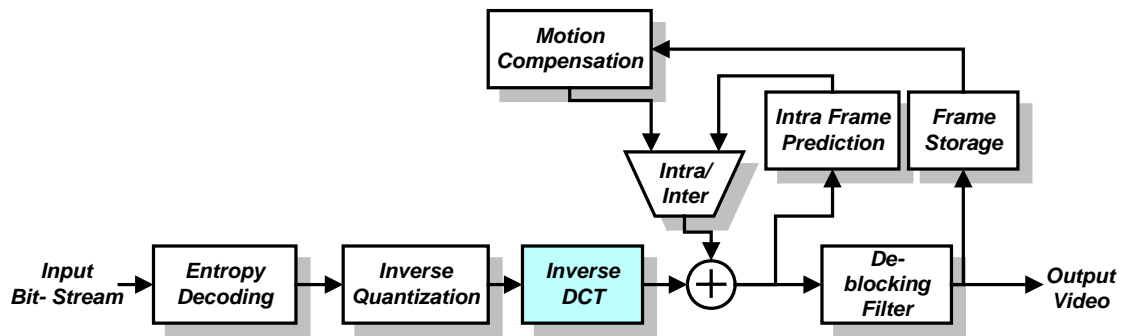


Fig.2.2 IDCT in H.264 Decoder

2.2 Overview of MPEG Compression Algorithm

The Moving Picture Experts Group (MPEG) standards determine two algorithms in implementing the video compression, First, Block based motion compensation is the temporal redundancy for reduce. Second, DCT compression is applied for spatial domain information.

2.2.1 Temporal Redundancy Reduction

The MPEG standard defines three types of pictures for motion compensation, they are, intra coded picture (I-Picture), predictive coded picture (P-Picture), and bi-directional predictive picture (B-Picture). The brief descriptions are listed below and the figure is in Fig.2.3.

I-Picture : Intra coded picture is without refer to other pictures, and it supposes an access points to the random access. And then, I-Picture offers moderate compression.

P-Picture : Predictive coded picture use the past I-Picture or P-Picture for motion compensation. The compression efficiency of P-Picture is better than I-Picture, from [5], I-Picture is three times longer than P-Picture.

B-Picture : Bi-directional predictive picture use the past I-Picture and future P-Picture for motion compression, The compression efficiency is the highest than I-Picture and P-Picture.

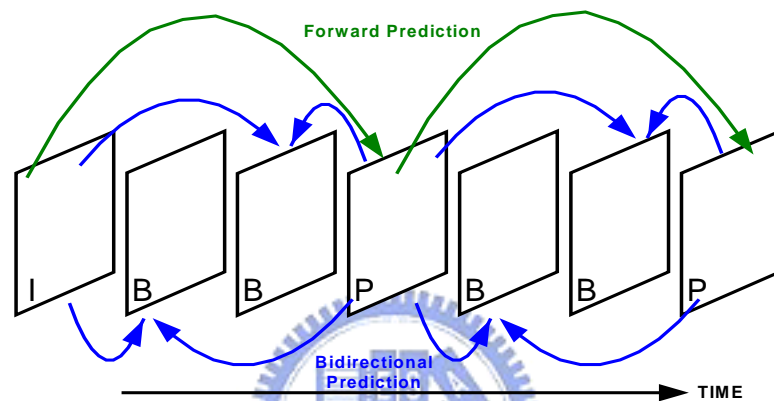


Fig.2.3 The Temporal Picture Structure

2.2.2 Spatial Redundancy Reduction

Both the still-image and prediction-error signals have a very high degree of spatial redundancy. The redundancy reductions techniques usable to this effect are many, but because of the block-based nature of the motion compression process, block based techniques are preferred. A frame is first divided into 8x8 blocks of pixels, and the two dimensional DCT is then applied independently on each block. This operation results in an 8x8 block of DCT coefficients in which most of the energy in the original block is typically concentrated in a few low frequency coefficients. A quantizer is applied to each DCT coefficients that sets many of them to zero. This quantization is responsible

for a lossy nature of the compression. Compression is achieved by transmitting only the coefficients that survive the quantization operation and by entropy coding their locations and amplitudes.

2.2.3 The Process of Decoding for MPEG-2

The MPEG-2 standard [6] defines the decoding process, and the mean is not the decoder, the designers and manufacturers can develop their own architecture and apply different algorithms to achieve such decoding process. The decoding process defined in MPEG-2 standards is showed in Fig.2.4. The input data is the first variable-length decoded. Since the data from VLD is zig-zag (Fig.2.5) scanned by the encoder, so the inverse scan module will reconstruct the one-dimension data stream into two-dimension matrix. This two-dimension matrix is then inverse quantized to obtain DCT coefficients. Note that the intra and inter block data will need different inverse quantization processes. The IDCT module transforms the coefficients into image data. The motion compensation module processes these image data together with motion vectors from VLD to form the decoded data. After proper filtering and transform, the image data are sent to display on the monitor or television.

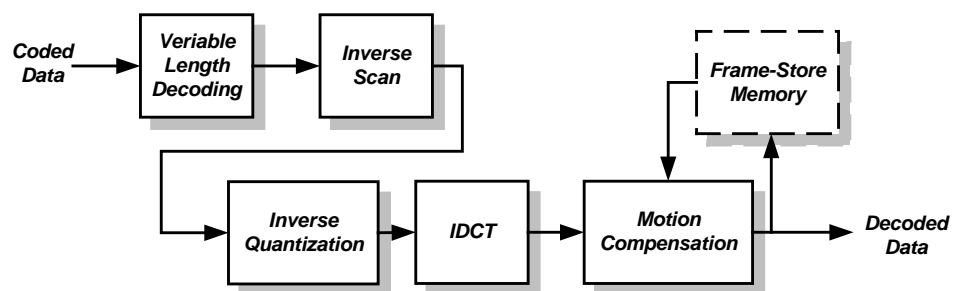


Fig.2.4 MPEG-2 Decoding Process

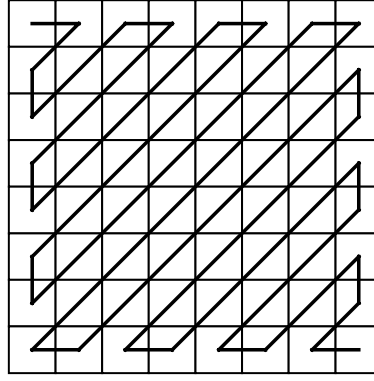


Fig.2.5 Zig-zag Scan Order

2.3 Algorithms of Inverse Discrete Cosine Transform

The $N \times N$ 2-D DCT is defined as following:

$$F(u, v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right] \quad (2.1)$$

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & u, v = 0 \\ 1 & u, v = 1, 2, \dots, N-1 \end{cases}$$

Where $f(x, y)$ is the pixel data, $F(x, y)$ is the transform coefficients, and $x, y, u, v = 0, 1, 2, \dots, N-1$

The $N \times N$ 2-D IDCT is defined as following:

$$f(x, y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right] \quad (2.2)$$

Where $C(u) = \sqrt{\frac{1}{2}}$ and $C(v) = 1$ for $k \neq 0$ and $x, y, u, v = 0, 1, 2, \dots, N-1$

For 2-D $N \times N$ IDCT, N^4 multipliers and N^4 adders are needed. That means that 2-D IDCT of an 8×8 block needs $2 \times 8^4 = 4096$ multiplications and additions to complete this IDCT transform. It is not feasible to implement IDCT using so much multiplications and additions, because it is very expensive. Until now, many algorithms have been

finding the method to reduce the amount of multiplications and additions, especially the multiplication. It needs large area and computation time in the chip.

2.3.1 Direct Computation of Two Dimension DCT

The algorithm to compute DCT directly was proposed by Chen, Smith and Tralick [7] the algorithm is explained listed below.

The DCT of an $N \times 1$ matrix form is

$$F = \left(\frac{2}{N} \right)^{\frac{1}{2}} A_N f \quad (2.3)$$

Where A_N is an $N \times N$ transform matrix for DCT, and it can be represented in a recursive form.

$$A_N = \begin{bmatrix} A_{\frac{N}{2}} & 0 \\ 0 & R_{\frac{N}{2}} \end{bmatrix} B_N \quad (2.4)$$

$$B_N = \begin{bmatrix} I_{\frac{N}{2}} & \overline{I_{\frac{N}{2}}} \\ \overline{I_{\frac{N}{2}}} & -I_{\frac{N}{2}} \end{bmatrix}$$



where $R_N = c(k) \cos \frac{(2j+1)(2k+1)\pi}{2N} \quad j, k = 0, 1, 2 \dots N-1$

The below diagram (Fig.2.6) signal flow graph for $N=8$ using sparse matrix direct computation algorithm.

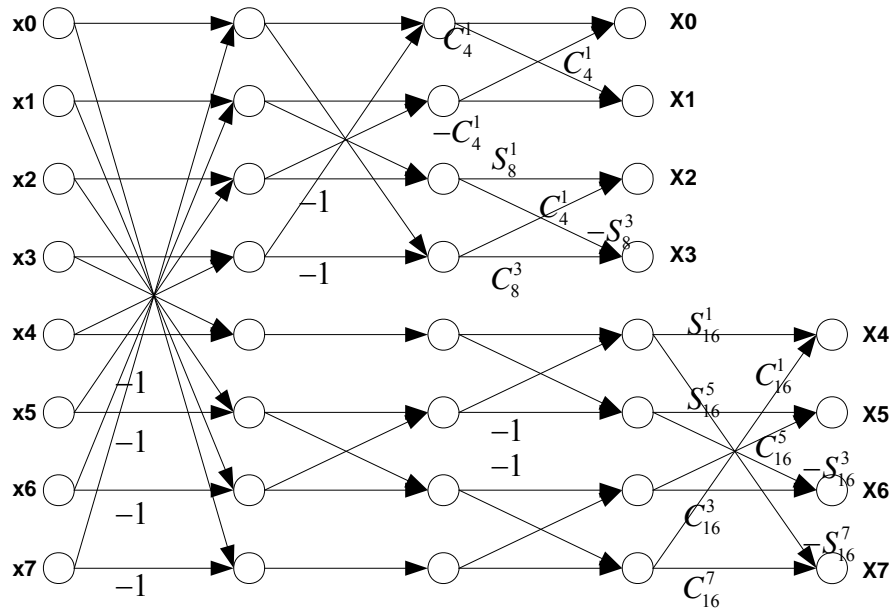


Fig.2.6 Signal Flow Graph of Direct Computation Algorithm for N=8

$$\begin{aligned}
 S_N^i &= \sin\left(\frac{iP_i}{N}\right) \\
 C_N^i &= \cos\left(\frac{iP_i}{N}\right)
 \end{aligned}
 \tag{2.5}$$



The algorithm requires

$$\begin{aligned}
 \frac{3}{2}(\log_2 N - 1) + 2 & \text{ real additions} \\
 N \log_2 N - \frac{3N}{2} + 4 & \text{ real multiplications}
 \end{aligned}
 \tag{2.6}$$

If $N = 8$, this algorithm needs 16 multiplications, base on cost down target, it still needs more multiplications for implementation.

2.3.2 Parallel Implementations

Cho and Lee [9] introduced the architecture that can be executed on the modified DFT architecture with $(N + 1)$ PE's. The relationship developed in this algorithm will be used later in the derivation of the prime factor DCT algorithm.

Bayoumi *et al.* [10] proposed a systolic array for computing the DFT based on the RNS (residue number system). Fig.2.7 depicts the architecture for 5-point DFT. From Fig.2.7, one can see that $(N - 1)$ basic PE's are required to compute N-point DFT. Each PE of the array performs the function shown in Fig.2.8.

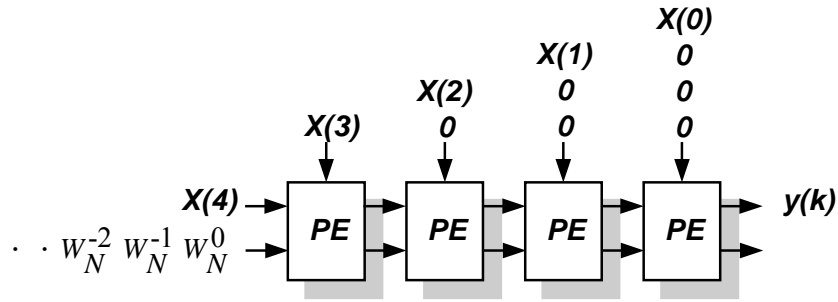


Fig.2.7 The Example of Bayoumi's [10] Architecture for 5 Point DFT

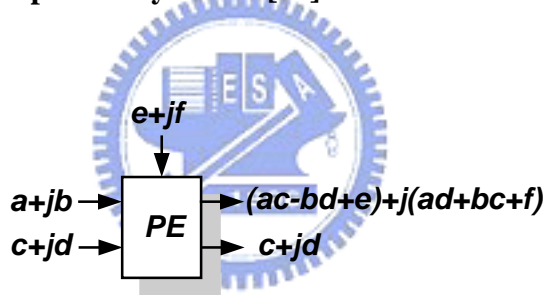


Fig.2.8 The Function of Each PE

N-point DFT can also be executed on this systolic array. Let the input data sequence be $\{X(n), n = 0, 1, \dots, N-1\}$, and the DFT of $X(n)$ be $\{Y(k), k = 0, 1, \dots, N-1\}$. Then the DFT is given by the following relation :

$$Y(k) = \sum_{n=0}^{N-1} X(n)W_N^{nk} \quad (2.7)$$

where

$$W_n = e^{-j\frac{2\pi}{N}}, \quad k = 0, 1, \dots, N-1$$

In EQ.2.7, skip a scale factor $\frac{1}{N}$ for convenience. The architecture proposed in [10], but it requires reverse input order. In most cases, natural order input is preferred since otherwise unnecessary delay and memory are required. In order to have architecture for natural order input, simple modification is needed. Let us denote $A(k)$ as the output when input data sequence to this architecture is in natural order and the kernel input is conjugated, i.e., the input is $\{W_N^0, W_N^{-1}, W_N^{-2}, \dots, W_N^{-(N-1)}\}$, Then instead of $\{W_N^0, W_N^1, W_N^2, \dots, W_N^{(N-1)}\}$. Then $A(k)$ can be expressed as

$$A(k) = \sum_{n=0}^{N-1} X(N-n-1)W_N^{-nk} \quad (2.8)$$

It is shown that the relationship between $A(k)$ and $Y(k)$ is

$$A(k) = W_N^k Y(k) \quad (2.9)$$

Thus, by connecting one additional basic PE to the N-point systolic array, as shown in Fig.2.10, we can obtain $W_N^{-k}A(k)$ at the output. Fig.2.10 depicts the modified DFT architecture for natural order input. Now we shall focus on N-point DCT, which can be executed on this modified DFT architecture for input in natural order. The DCT relationship is given in EQ.2.10.

$$Y(k) = c(k) \sum_{n=0}^{N-1} X(n) \cos \left[\frac{(2n+1)k\pi}{2N} \right] \quad (2.10)$$

where

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & ; k = 0 \\ 1 & ; otherwise \end{cases}$$

Here we also neglect the scale factor $\frac{1}{N}$ until the end for clarity.

Instead of computing EQ.2.10 directly, one can see that the DCT can also be obtained by the indirect computations of EQ.2.11 and EQ.2.12.

$$T(k) = \sum_{n=0}^{N-1} X(n) U_N^{nk} \quad (2.11)$$

where

$$U_N = e^{j\frac{\pi}{N}}, \quad k = 0, 1, \dots, N-1$$

Then $Y(k)$ can be expressed as

$$Y(k) = c(k) \operatorname{Re} \left\{ e^{j\frac{k\pi}{2N}} T(k) \right\} \quad (2.12)$$

On the systolic architecture of Fig.2.10, if we input $\{U_N^0, U_N^{-1}, U_N^{-2}, \dots, U_N^{-(N-1)}\}$ instead of $\{W_N^0, W_N^{-1}, W_N^{-2}, \dots, W_N^{-(N-1)}\}$, then the output at the right end of the architecture is

$$B(k) = U_N^{-k} \sum_{n=0}^{N-1} X(N-n-1) U_N^{-nk} \quad (2.13)$$

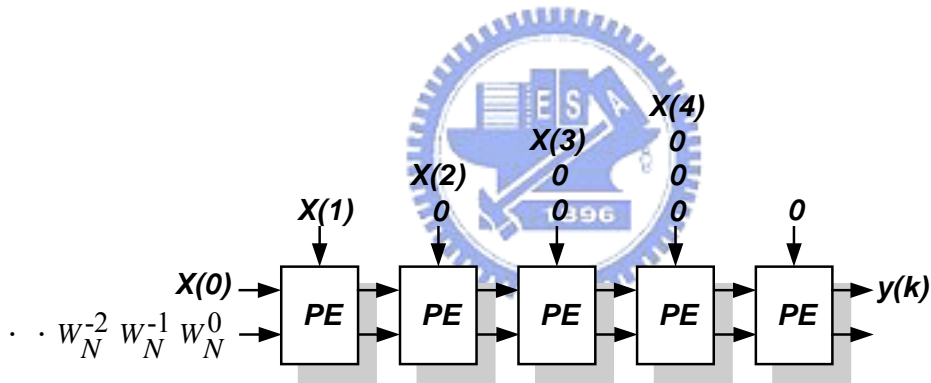


Fig.2.9 The Modified Systolic Architecture for 5 Point DFT

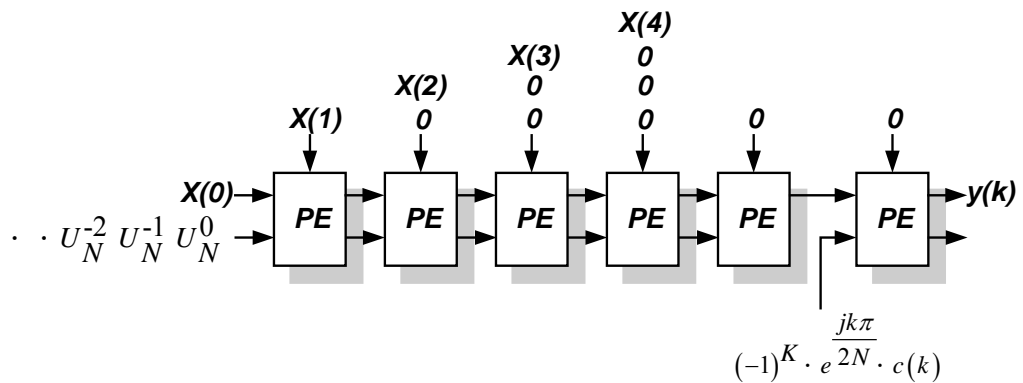


Fig.2.10 Systolic Architecture for 5 Point DFT

which is equivalent to

$$B(k) = U_N^{-Nk} T(k) \quad (2.14)$$

Since $U_N^{-Nk} = (-1)^k$, the output of the system is equal to $T(k)$ if k is even; and it is equal to $-T(k)$ if k is odd.

Thus, if we connect a processor that multiplies $(-1)^k e^{j\frac{k\pi}{2N}} c(k)$ to $B(k)$, then DCT is finally obtained.

The architecture for the case of 5-point DCT is shown in Fig.2.10. Thus, the total number of PE's required for N -point DCT is $(N + 1)$, if one prefers to arrange the input data in reverse order, the number of PE's for N -point DCT would be N . Since DST (discrete sine transform) is similarly defined as DCT, we can also obtain the DST on this architecture with slight modifications.



2.4 Paper Reference

This section will introduce the DCT/ICDT architecture and methods in the state-of-art-works, lists the main methods and show the architecture diagram of them.

In D.W Kim [14] proposal, the architecture uses hardwired DA method, radix-2 multi-bit coding methods, Fig.2.11 shows the processing element for IDCT even and odd matrix.

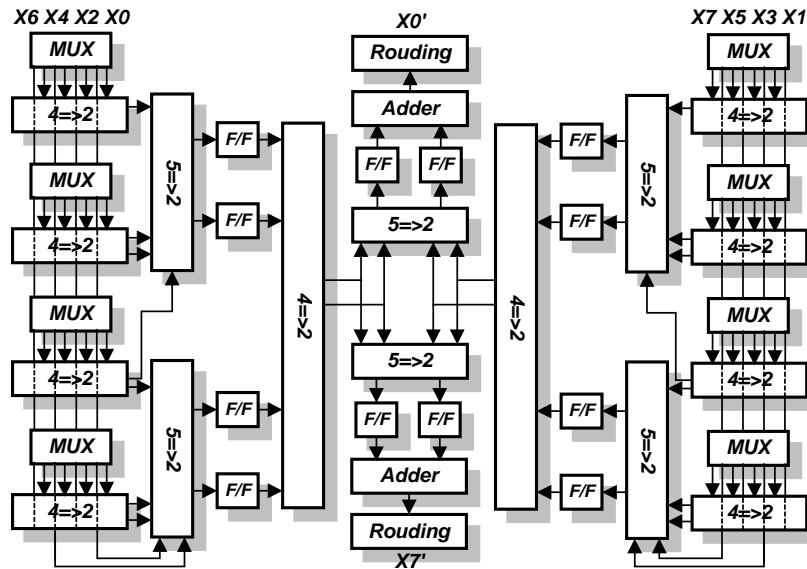
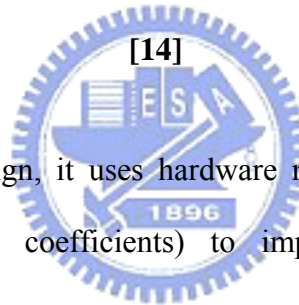


Fig.2.11 Processing Element for IDCT Even and Odd Matrix from D.W Kim



In A. Madisetti [15] design, it uses hardware multiplications and signed digit representation (12bits cosine coefficients) to implement. Fig.2.12 shows this architecture.

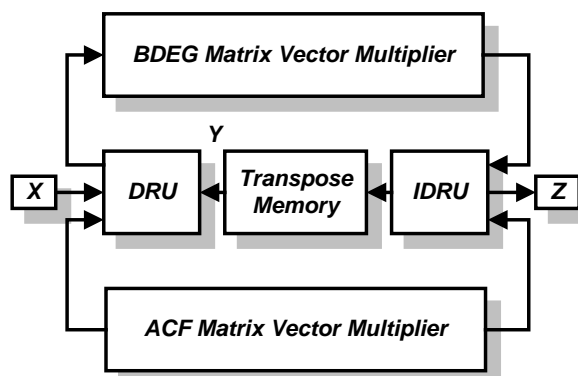


Fig.2.12 A. Madisetti's [15] Process Architecture

This architecture of the chip consists data recorder unit (DRU), two matrix-vector

multiplier units, inverse data recorder unit (IDRU), and transpose memory. Fig.2.13 to Fig.2.15 show the details of the [15] process architecture.

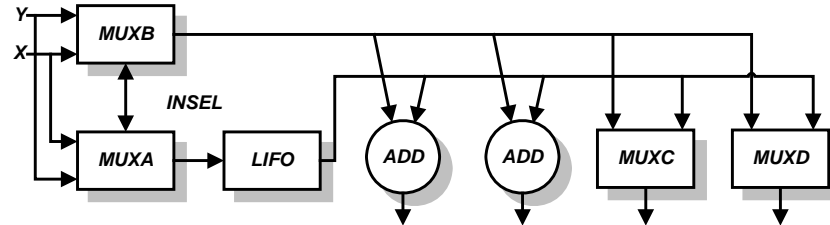


Fig.2.13 Data Recorder Unit (DRU) from [15]

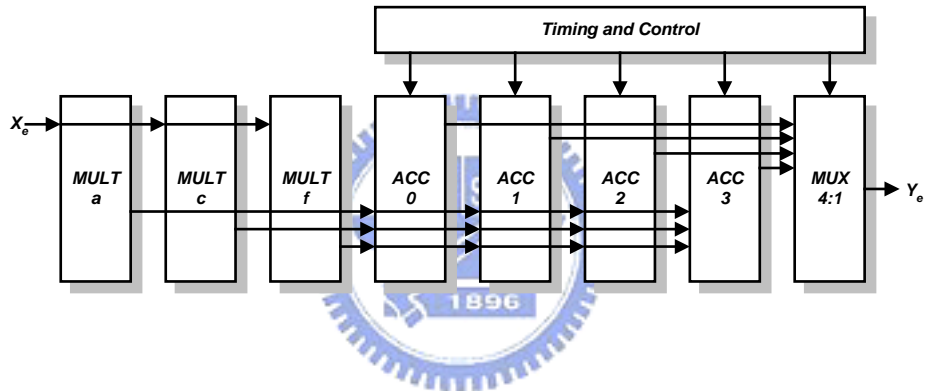


Fig.2.14 ACF Matrix-Vector Multiply Unit from [15]

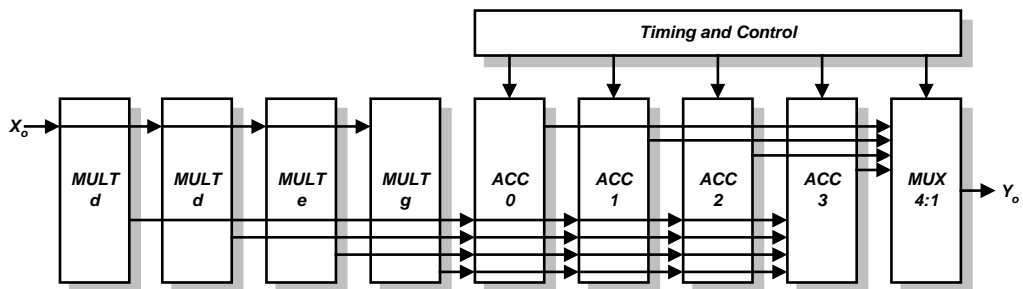


Fig.2.15 BDEG Matrix-Vector Multiply Unit from [15]

J.I Guo's [16] design uses hardwired multiplications, cyclic convolution, signed

digit representation (14bits cosine coefficients), and common sub-expression sharing methods, the architecture is shown in Fig.2.16.

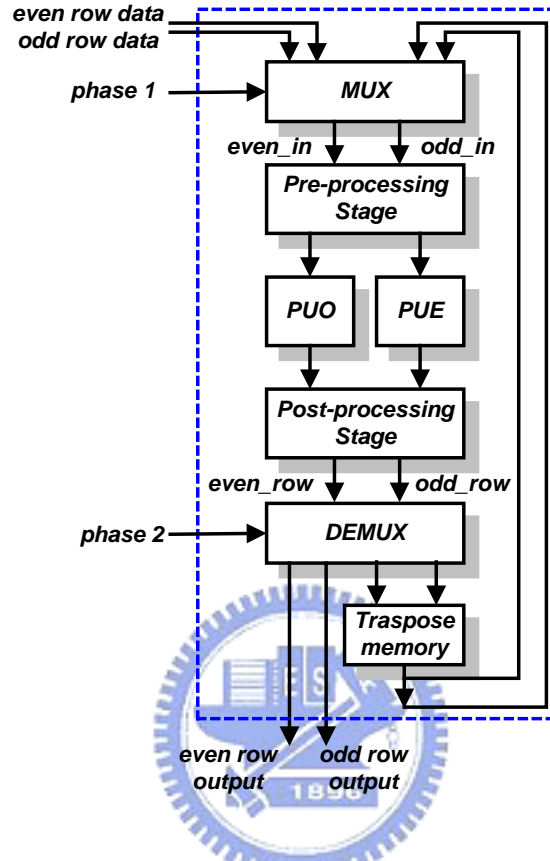


Fig.2.16 J.I Guo's [16] Architecture

2.5 The Proposed Algorithm

From [8], an algorithm is described as follows:

$$\text{The IDCT } x(k) = \sum_{n=0}^{N-1} C(n) X(n) \cos \left[\frac{(2k+1)n\pi}{2N} \right] \quad (2.15)$$

$$\text{Where } C(n) \text{ is } \begin{cases} 1 & n = 0 \\ \sqrt{2} & n = 1, 2, \dots, N-1 \\ 1 & \end{cases}$$

$$\text{Let } C_{2N}^{(2k+1)n} = \cos \frac{(2K+1)n\pi}{2N} \quad (2.16)$$

and the IDCT can be expressed as

$$x(k) = \sum_{n=0}^{N-1} \widehat{X}(n) C_{2N}^{(2k+1)n} \quad k = 0, 1, 2, \dots, N-1 \quad (2.17)$$

$$\text{Where } \widehat{X}(n) = e(n) X(n) \quad (2.18)$$

If N is even, $x(k)$ separates even and odd values, the formulas are listed below:

$$x(k) = g(k) + h'(k) \quad (2.19)$$

$$x(N-1-k) = g(k) - h'(k) \quad (2.20)$$

$$\text{Where } g(k) = \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n) C_{2N}^{(2k+1)2n} \quad \text{and} \quad h'(k) = \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)(2n+1)} \quad (2.21)$$

Then $g(k), k = 0, 1, 2, \dots, \frac{N}{2}-1$ makes a $\frac{N}{2}$ point IDCT. Rewrite $h'(k)$:

$$h'(k) = \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)2n} \quad (2.22)$$

is another $\frac{N}{2}$ point IDCT, and

$$2C_{2N}^{(2k+1)} C_{2N}^{(2k+1)(2n+1)} = C_{2N}^{(2k+1)2n} + C_{2N}^{(2k+1)2n+1} \quad (2.23)$$

Form EQ.2.22 and EQ.2.23 the equation becomes

$$2C_{2N}^{(2k+1)} h'(k) = \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)2n} + \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)2n+1} \quad (2.24)$$

$$\text{if } \widehat{X}(-1) = 0, \quad \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)2(n+1)} = \sum_{n=0}^{\frac{N-1}{2}} \widehat{X}(2n+1) C_{2N}^{(2k+1)2n} \quad (2.25)$$

$$\text{EQ.2.25 rewrite to } 2C_{2N}^{(2k+1)} h'(k) = \sum_{n=0}^{\frac{N-1}{2}} (\widehat{X}(2n+1) + \widehat{X}(2n-1)) C_{2N}^{(2k+1)2n} \quad (2.26)$$

Then define

$$G(n) = \widehat{X}(2n), H(n) = \widehat{X}(2n+1) + \widehat{X}(2n-1) \quad n = 0, 1, 2, \dots, \frac{N}{2} - 1 \quad (2.27)$$

$$h(k) = \sum_{n=0}^{\frac{N}{2}-1} G(n) C_{2\left(\frac{N}{2}\right)}^{(2k+1)n} \quad k = 0, 1, 2, \dots, \frac{N}{2} - 1, \quad \text{and} \quad g(k) = \sum_{n=0}^{\frac{N}{2}-1} H(n) C_{2\left(\frac{N}{2}\right)}^{(2k+1)n} \quad (2.28)$$

Finally the formulas are

$$x(k) = g(k) + \frac{1}{2C_{2N}^{(2k+1)}} h(k) \quad (2.29)$$

$$x(N-1-k) = g(k) - \frac{1}{2C_{2N}^{(2k+1)}} h(k) \quad k = 0, 1, 2, \dots, \frac{N}{2} - 1 \quad (2.30)$$

From the upper process, we can use two $\frac{N}{2}$ points IDCT to calculate N -point

IDCT. Fig.2.17 is the 8-point IDCT flow graph.

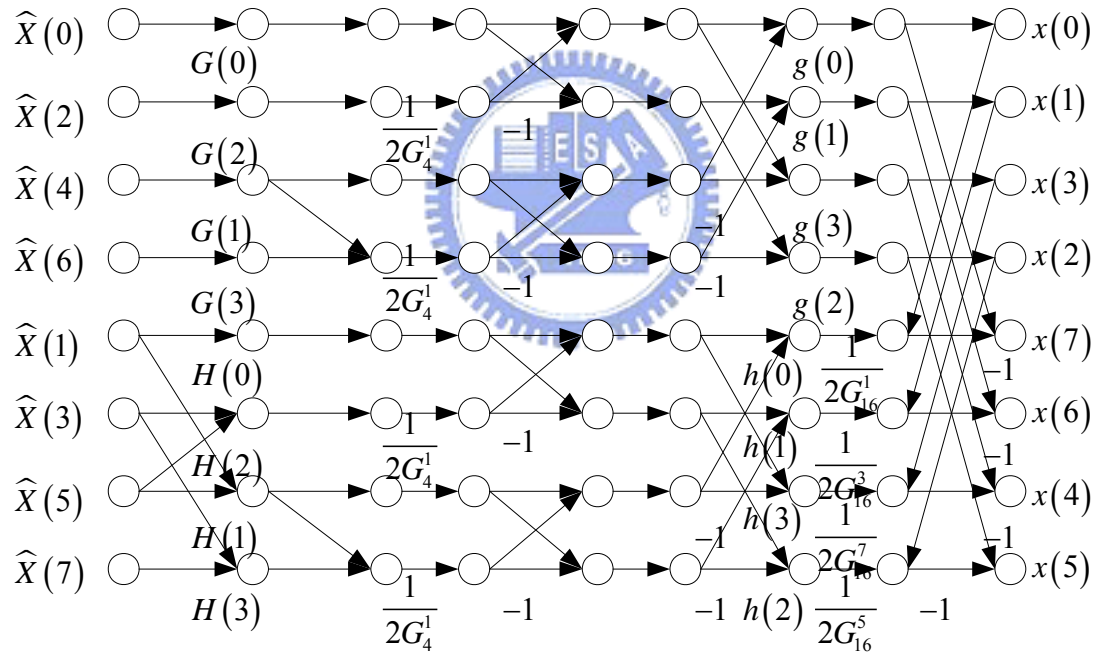


Fig.2.17 The Flow Graph of FCT for N=8

2.5.1 IDCT of MPEG-2

The one dimension 8-point IDCT formula is listed in EQ.2.31 and it is expressed by 8×8 matrix shown in Fig.2.18. The coefficients are listed below.

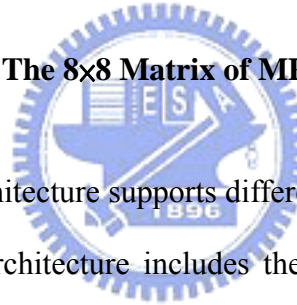
$$x(k) = \frac{1}{2} \sum_{n=0}^7 C(n) X(n) \cos \left[\frac{(2k+1)n\pi}{16} \right]$$

$$C(n) = \begin{cases} \frac{1}{\sqrt{2}} & n=0 \\ 1 & n=1 \sim 7 \end{cases} \quad (2.31)$$

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} & x_{07} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\ x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\ x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\ x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\ x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & B & C & D & A & E & F & G \\ A & D & F & -G & -A & -B & -C & -E \\ A & E & -F & -B & -A & G & C & D \\ A & G & -C & -E & A & D & -F & -B \\ A & -G & -C & E & A & -D & -F & B \\ A & -E & -F & B & -A & -G & C & -D \\ A & -D & F & G & -A & B & -C & E \\ A & -B & C & -D & A & -E & F & -G \end{bmatrix} \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} & X_{04} & X_{05} & X_{06} & X_{07} \\ X_{10} & X_{11} & X_{12} & X_{13} & X_{14} & X_{15} & X_{16} & X_{17} \\ X_{20} & X_{21} & X_{22} & X_{23} & X_{24} & X_{25} & X_{26} & X_{27} \\ X_{30} & X_{31} & X_{32} & X_{33} & X_{34} & X_{35} & X_{36} & X_{37} \\ X_{40} & X_{41} & X_{42} & X_{43} & X_{44} & X_{45} & X_{46} & X_{47} \\ X_{50} & X_{51} & X_{52} & X_{53} & X_{54} & X_{55} & X_{56} & X_{57} \\ X_{60} & X_{61} & X_{62} & X_{63} & X_{64} & X_{65} & X_{66} & X_{67} \\ X_{70} & X_{71} & X_{72} & X_{73} & X_{74} & X_{75} & X_{76} & X_{77} \end{bmatrix}$$

$$A = \cos \frac{\pi}{4}; B = \cos \frac{\pi}{16}; C = \cos \frac{\pi}{8}; D = \cos \frac{\pi}{16}; E = \cos \frac{5\pi}{16}; F = \cos \frac{3\pi}{8}; G = \cos \frac{7\pi}{16}$$

Fig.2.18 The 8x8 Matrix of MPEG-2 IDCT



Because the proposed architecture supports different standards, and they have 8x8 and 4x4 matrix, to use one architecture includes these different matrix, 8x8 matrix need transport to 4x4 architecture, and this transport algorithm is showed in Fig.2.19.

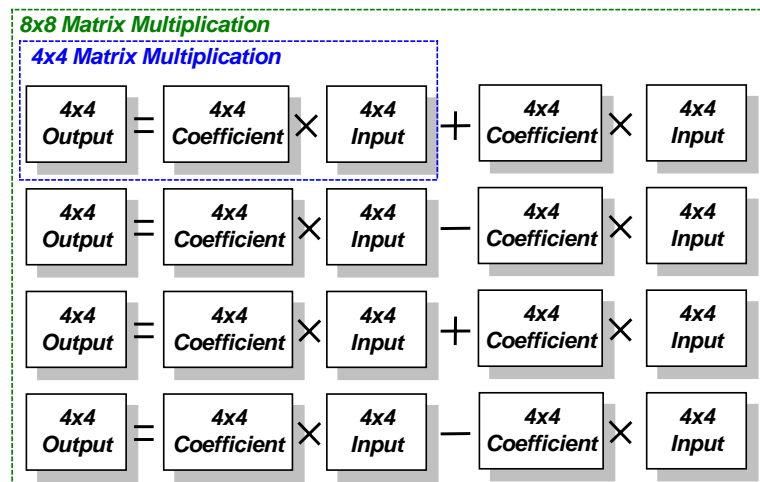


Fig.2.19 The Transport for 8x8 Matrix

From Fig.2.19, because the coefficient matrix is symmetrical, it can be separated into two matrix-vectors (an 8×8 matrix into two 4×4 matrices) as shown in Fig.2.20.

$$\begin{aligned}
 & \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & C & A & F \\ A & F & -A & -C \\ A & -F & -A & C \\ A & -C & A & -F \end{bmatrix} \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{40} & X_{41} & X_{42} & X_{43} \\ X_{60} & X_{61} & X_{62} & X_{63} \end{bmatrix} + \begin{bmatrix} A & D & E & G \\ D & -G & -B & -E \\ E & -B & G & D \\ G & -E & D & -B \end{bmatrix} \begin{bmatrix} X_{10} & X_{11} & X_{12} & X_{13} \\ X_{30} & X_{31} & X_{32} & X_{33} \\ X_{50} & X_{51} & X_{52} & X_{53} \\ X_{70} & X_{71} & X_{72} & X_{73} \end{bmatrix} \\
 & \begin{bmatrix} x_{70} & x_{71} & x_{72} & x_{73} \\ x_{60} & x_{61} & x_{62} & x_{63} \\ x_{50} & x_{51} & x_{52} & x_{53} \\ x_{40} & x_{41} & x_{42} & x_{43} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & C & A & F \\ A & F & -A & -C \\ A & -F & -A & C \\ A & -C & A & -F \end{bmatrix} \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{40} & X_{41} & X_{42} & X_{43} \\ X_{60} & X_{61} & X_{62} & X_{63} \end{bmatrix} - \begin{bmatrix} A & D & E & G \\ D & -G & -B & -E \\ E & -B & G & D \\ G & -E & D & -B \end{bmatrix} \begin{bmatrix} X_{10} & X_{11} & X_{12} & X_{13} \\ X_{30} & X_{31} & X_{32} & X_{33} \\ X_{50} & X_{51} & X_{52} & X_{53} \\ X_{70} & X_{71} & X_{72} & X_{73} \end{bmatrix} \\
 & \begin{bmatrix} x_{04} & x_{05} & x_{06} & x_{07} \\ x_{14} & x_{15} & x_{16} & x_{17} \\ x_{24} & x_{25} & x_{26} & x_{27} \\ x_{34} & x_{35} & x_{36} & x_{37} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & C & A & F \\ A & F & -A & -C \\ A & -F & -A & C \\ A & -C & A & -F \end{bmatrix} \begin{bmatrix} X_{04} & X_{05} & X_{06} & X_{07} \\ X_{24} & X_{25} & X_{26} & X_{27} \\ X_{44} & X_{45} & X_{46} & X_{47} \\ X_{64} & X_{65} & X_{66} & X_{67} \end{bmatrix} + \begin{bmatrix} A & D & E & G \\ D & -G & -B & -E \\ E & -B & G & D \\ G & -E & D & -B \end{bmatrix} \begin{bmatrix} X_{14} & X_{15} & X_{16} & X_{17} \\ X_{34} & X_{35} & X_{36} & X_{37} \\ X_{54} & X_{55} & X_{56} & X_{57} \\ X_{74} & X_{75} & X_{76} & X_{77} \end{bmatrix} \\
 & \begin{bmatrix} x_{74} & x_{75} & x_{76} & x_{77} \\ x_{64} & x_{65} & x_{66} & x_{67} \\ x_{54} & x_{55} & x_{56} & x_{57} \\ x_{44} & x_{45} & x_{46} & x_{47} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & C & A & F \\ A & F & -A & -C \\ A & -F & -A & C \\ A & -C & A & -F \end{bmatrix} \begin{bmatrix} X_{04} & X_{05} & X_{06} & X_{07} \\ X_{24} & X_{25} & X_{26} & X_{27} \\ X_{44} & X_{45} & X_{46} & X_{47} \\ X_{64} & X_{65} & X_{66} & X_{67} \end{bmatrix} - \begin{bmatrix} A & D & E & G \\ D & -G & -B & -E \\ E & -B & G & D \\ G & -E & D & -B \end{bmatrix} \begin{bmatrix} X_{14} & X_{15} & X_{16} & X_{17} \\ X_{34} & X_{35} & X_{36} & X_{37} \\ X_{54} & X_{55} & X_{56} & X_{57} \\ X_{74} & X_{75} & X_{76} & X_{77} \end{bmatrix}
 \end{aligned}$$

Fig.2.20 Separate 8×8 MPEG-2 IDCT into 4×4 Matrix

2.5.2 IDCT of H.264 High Profile

The 8×8 matrix of H.264 high profile IDCT is shown in Fig.2.21. In order to carry out the matching separation of an 8×8 matrix into two 4×4 matrices of MPEG-2 IDCT, the column placement of multiplicand and multiplier need to be exchanged. Specifically, columns #5 and #8, #6 and #7 in multiplicand matrix and rows #5 and #8, #6 and #7 in multiplier matrix each needs to be exchanged (the changed result is shown in Fig.2.22), the separated matrix is shown in Fig.2.23

2.5.3 IDCT of H.264 Baseline

The H.264 baseline IDCT is a 4x4 matrix as shown in Fig.2.24, and it can be calculated by the 4x4 architecture directly. In other words, it does not need to undergo any modification.

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

Fig.2.24 The 4x4 Matrix of H.264 Baseline IDCT



Chapter 3

Architecture Overview

In this chapter, we will propose the IDCT architecture, input and output interface. In matrix multiplication calculation block, one dimension systolic array and re-arranging input data flow to the matrix multiplication will be introduced. To match the system requirement and the replacing of multiplication, we introduce some methods to optimize this architecture.

This chapter is organized as follows. In section 3.1, the proposed prototype architecture will be introduced. In section 3.2, we present the overview of one dimension systolic array and modified the dataflow to calculate matrix multiplication. In section 3.3, consider the system requirement and motion compression, we modify the prototype architecture to match the system request, and the method to optimize the addition amount, then new architecture will be given.

3.1 Propose System Architecture of IDCT

The IDCT operating procedure is shown in Fig.3.1. It consists of two individual one dimension IDCT and one transpose process which is used to combines this two one dimension IDCT.

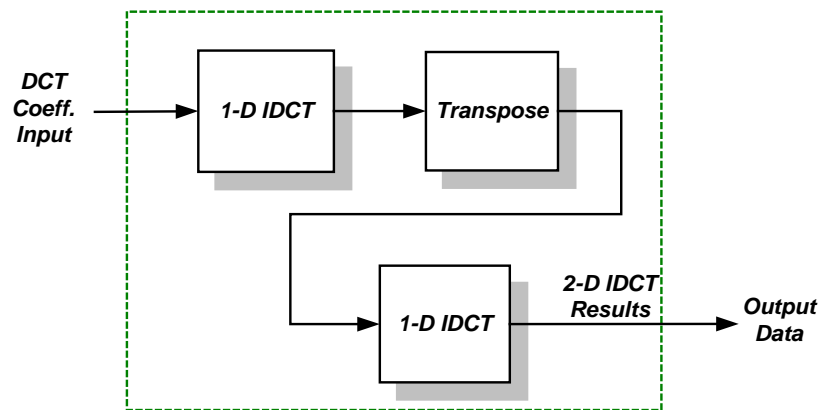


Fig.3.1 IDCT Operating Procedure

About the input interface connecting from inverse quantization, an input buffer is inserted after inverse quantization. It saves the DCT coefficients calculated by inverse quantization, in 4×4 matrix calculation case, this buffer transfers four 16-bit coefficients and output to IDCT. Another four 16-bit coefficients output controlled by MODE control signal of IDCT block and is enable in 8×8 matrix calculation. The input interface is shown in Fig.3.2.

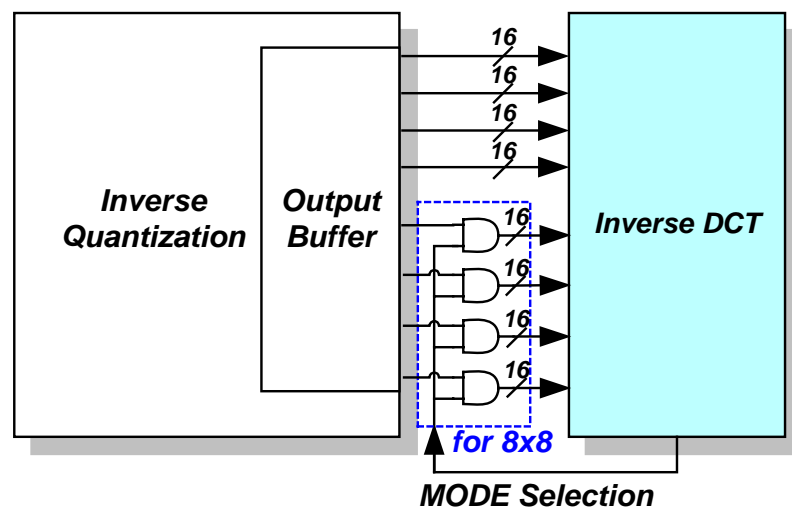


Fig.3.2 IDCT Block Input Interface

The block diagram of the prototype of IDCT architecture is shown in Fig.3.3. Four 16-bit coefficients output from inverse quantization as an input of IDCT.

If the case in 8x8 standard, they will be divided into two 4x4 matrix blocks. Because this architecture supports three multimedia video standards namely MPEG-2, H.264 high profile, and H.264 baseline, the coefficient matrix (multiplicand matrix) is defined by the two bits signal named “MODE”. The selected MODE outputs the suitable coefficients matrix using in matrix multiplication calculation, then the addition and the subtraction processes are executed to obtain the 4x4 matrix multiplication solutions. Finally, the obtained four 4x4 results are combined to generate the 8x8 matrix. About matrix multiplication calculation block, next section will introduce this method.

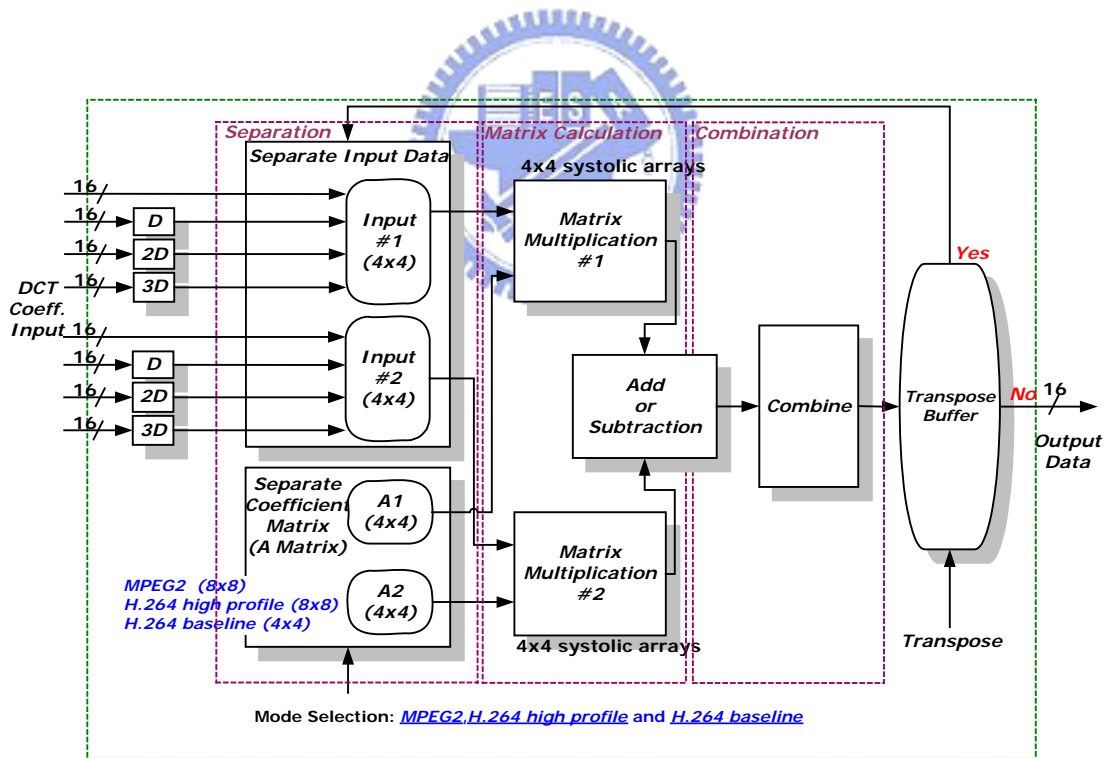


Fig.3.3 Block Diagram of Prototype IDCT Architecture

3.2 Overview of One Dimension Systolic Array

For matrix multiplication calculation, one dimension systolic array is used to perform matrix multiplication after it has been modified by input data flow. One dimension systolic has the characteristics of "regularity" and "inherent parallelism" in input and output data flow. Besides, it also provides easier architecture design and layout. Because the result of IDCT needs to add the result of the motion compensation where the latter is always the bottleneck of the multimedia calculation process. As a result, the IDCT design sacrifices some throughput to reduce the cost. So the method of one dimension systolic array is adopted. In [11], the one dimension systolic array used in the motion estimation; the data flow of the motion estimation is shown in Fig.3.4.

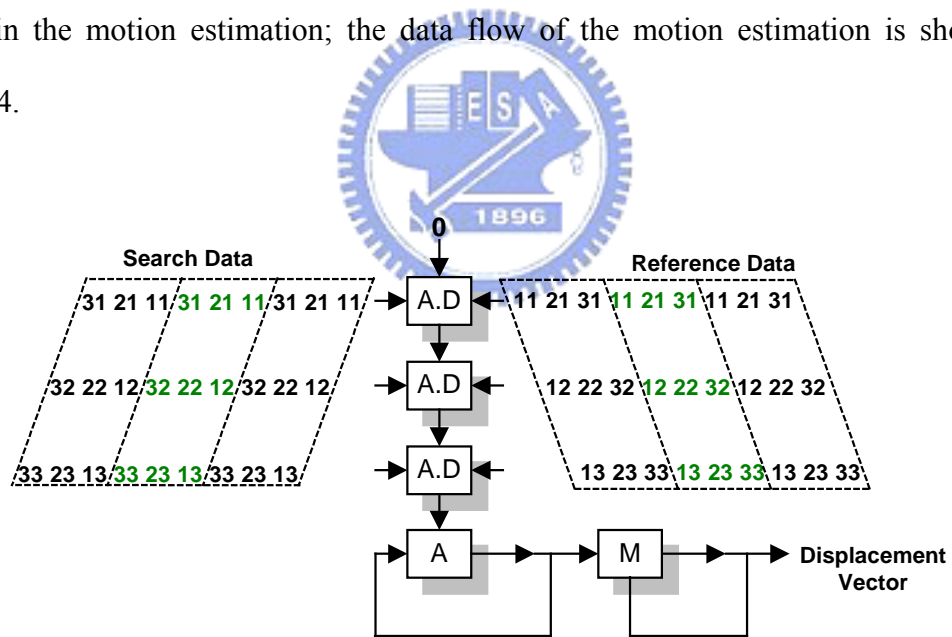


Fig.3.4 One-D Systolic Array Data Flow for Motion Estimation [11]

As Fig.3.5 shows, it can modify the multiplicand and multiplier data flow to perform matrix multiplication calculation, four processing elements (PE's) in it. Processing element #1 is a multiplier, and processing elements #2~#4 include

multiplier and adder. The data flow with example also listed in Fig.3.5.

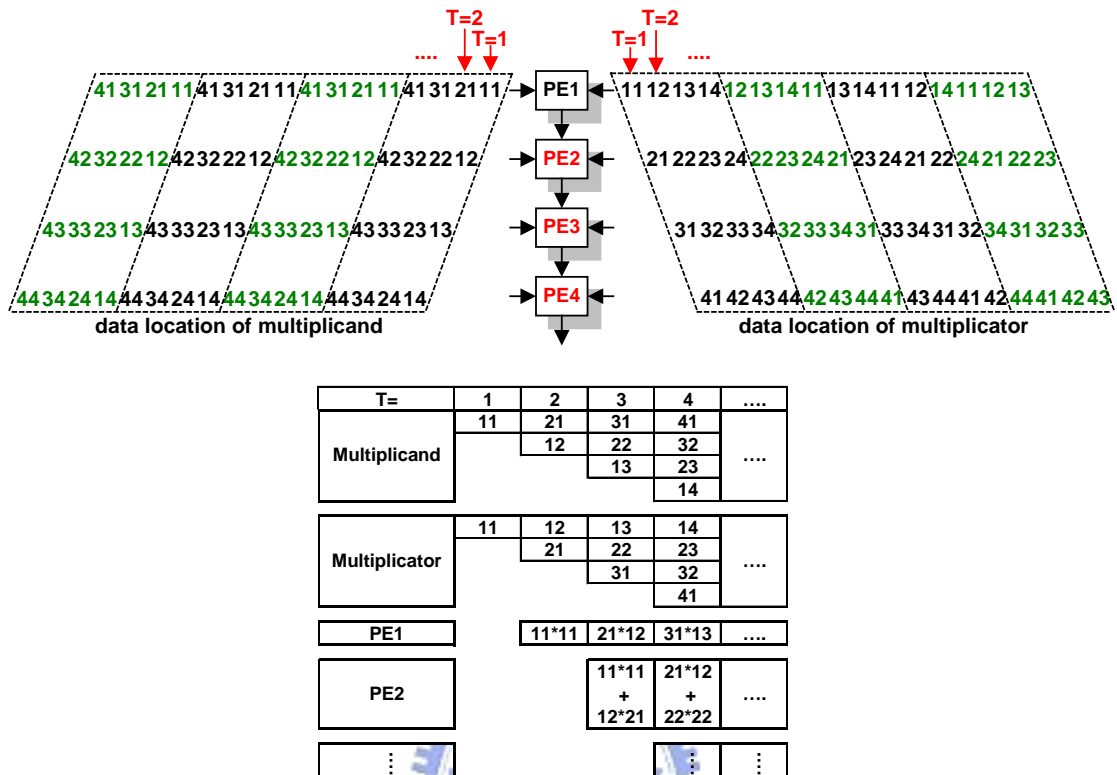


Fig.3.5 One-D Systolic Array Data Flow for 4x4 Matrix Multiplication

The estimated cycle counts of one dimension systolic array for performing 4x4 matrix multiplication is $N + (N \times N) = 20$ cycles ($N = 4$). The architecture is shown in Fig.3.6. Accessing addition and subtraction processes needs 2 cycles; therefore the two-D 8×8 matrix in the worst case of MPEG-2 needs $2 \times \{2 \times [N + (N \times N) + 1]\} = 84$ cycles. However, the cycle counts of motion compensation and system requirement need under 400 cycles per microblock in 4:2:0 standards, the two-D 8×8 matrix needs 84 cycles required by MPEG-2 is too slow for the system requirements and motion compensation produces the result.

cycle #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Multiplicand	a11	a21	a31	a41	a11	a21	a31	a41	a11	a21	a31	a41	a11	a21	a31	a41					
		a12	a22	a32	a42	a12	a22	a32	a42	a12	a22	a32	a42	a12	a22	a32	a42				
			a13	a23	a33	a43	a13	a23	a33	a43	a13	a23	a33	a43	a13	a23	a33	a43			
				a14	a24	a34	a44	a14	a24	a34	a44	a14	a24	a34	a44	a14	a24	a34	a44		
Multiplier	b11	b12	b13	b14	b12	b13	b14	b11	b13	b14	b11	b12	b14	b11	b12	b13					
		b21	b22	b23	b24	b22	b23	b24	b21	b23	b24	b21	b22	b24	b21	b22	b23				
			b31	b32	b33	b34	b32	b33	b34	b31	b33	b34	b31	b32	b34	b31	b32	b33			
				b41	b42	b43	b42	b43	b41	b43	b41	b43	b44	b41	b32	b44	b41	b42	b43		
PE1	a11*b11 a21*b12 a31*b13 a41*b14 a11*b12 a21*b13 a31*b14 a41*b11 a11*b13 a21*b14 a31*b11 a41*b12 a11*b14 a21*b11 a31*b12 a41*b13																				
PE2	a11*b11 a21*b12 a31*b13 a41*b14 a11*b12 a21*b13 a31*b14 a41*b11 a11*b13 a21*b14 a31*b11 a41*b12 a11*b14 a21*b11 a31*b12 a41*b13 a12*b21 a22*b22 a32*b23 a42*b24 a12*b22 a22*b23 a32*b24 a42*b21 a12*b23 a22*b24 a32*b21 a42*b22 a12*b24 a22*b21 a32*b22 a42*b23																				
PE3	a11*b11 a21*b12 a31*b13 a41*b14 a11*b12 a21*b13 a31*b14 a41*b11 a11*b13 a21*b14 a31*b11 a41*b12 a11*b14 a21*b11 a31*b12 a41*b13 a12*b21 a22*b22 a32*b23 a42*b24 a12*b22 a22*b23 a32*b24 a42*b21 a12*b23 a22*b24 a32*b21 a42*b22 a12*b24 a22*b21 a32*b22 a42*b23 a13*b31 a23*b32 a33*b33 a43*b34 a13*b32 a23*b33 a33*b34 a43*b31 a13*b33 a23*b34 a33*b31 a43*b32 a13*b34 a23*b31 a33*b32 a43*b33																				
PE4	a11*b11 a21*b12 a31*b13 a41*b14 a11*b12 a21*b13 a31*b14 a41*b11 a11*b13 a21*b14 a31*b11 a41*b12 a11*b14 a21*b11 a31*b12 a41*b13 a12*b21 a22*b22 a32*b23 a42*b24 a12*b22 a22*b23 a32*b24 a42*b21 a12*b23 a22*b24 a32*b21 a42*b22 a12*b24 a22*b21 a32*b22 a42*b23 a13*b31 a23*b32 a33*b33 a43*b34 a13*b32 a23*b33 a33*b34 a43*b31 a13*b33 a23*b34 a33*b31 a43*b32 a13*b34 a23*b31 a33*b32 a43*b33 a14*b41 a24*b42 a34*b43 a44*b44 a14*b42 a24*b43 a34*b44 a44*b41 a14*b43 a24*b44 a34*b41 a44*b42 a14*b44 a24*b41 a34*b42 a44*b43																				
value	c11	c22	c33	c44	c12	c23	c34	c41	c13	c24	c31	c42	c14	c21	c32	c43					

Fig.3.6 One-D Systolic Array Architecture for Matrix Multiplication

Regarding the throughput issue, the one dimension systolic array offers the elasticity required to solve this problem. It is able to add more hardware to increase the throughput, the worst case in IDCT is able to meet the system and motion compensation requirements. The data is listed in Table 3.1.

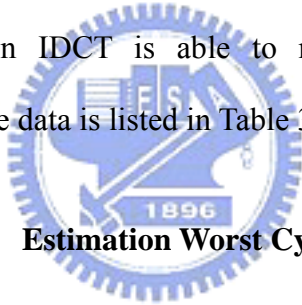


Table 3.1 Estimation Worst Cycle Counts

Standard	Size	Cycle Count (worst case)
		four PE's
MPEG-2	4x4 Matrix	$N+N^2$
	2-D 8x8 Matrix (for 4:2:0 system)	$2 \times \{2 \times [(N+N^2+1)]\}$

If the architecture increases more hardware, and need adjust the input data flow, the data flow is shown in Fig.3.7.

cycle #	1	2	3	4	5	6	7	8	9	10	11	12
Multipli cand	a11	a21	a31	a41	a11	a21	a31	a41				
	a11	a21	a31	a41	a11	a21	a31	a41				
		a12	a22	a32	a42	a12	a22	a32	a42			
		a12	a22	a32	a42	a12	a22	a32	a42			
		a13	a23	a33	a43	a13	a23	a33	a43			
		a13	a23	a33	a43	a13	a23	a33	a43			
			a14	a24	a34	a44	a14	a24	a34	a44		
			a14	a24	a34	a44	a14	a24	a34	a44		
Multipli cator	i11	i12	i13	i14	i13	i14	i11	i12				
	i12	i13	i14	i11	i14	i11	i12	i13				
		i21	i22	i23	i24	i23	i24	i21	i22			
		i21	i22	i23	i24	i21	i24	i21	i22	i23		
		i31	i32	i33	i34	i33	i34	i31	i32			
		i31	i32	i33	i34	i31	i34	i31	i32	i33		
			i41	i42	i43	i44	i43	i44	i41	i42	i32	
			i42	i43	i44	i41	i44	i41	i42	i43	i43	
PE1-1		a11*i11	a21*i12	a31*i13	a41*i14	a11*i13	a21*i14	a31*i11	a41*i12			
PE1-2		a11*i12	a21*i13	a31*i14	a41*i11	a11*i14	a21*i11	a31*i12	a41*i13			
PE2-1			a11*i11	a21*i12	a31*i13	a41*i14	a11*i13	a21*i14	a31*i11	a41*i12		
			a12*i21	a22*i22	a32*i23	a42*i24	a12*i23	a22*i24	a32*i21	a42*i22		
PE2-2			a11*i12	a21*i13	a31*i14	a41*i11	a11*i14	a21*i11	a31*i12	a41*i13		
			a12*i22	a22*i23	a32*i24	a42*i21	a12*i24	a22*i21	a32*i22	a42*i23		
PE3-1				a11*i11	a21*i12	a31*i13	a41*i14	a11*i13	a21*i14	a31*i11	a41*i12	
				a12*i21	a22*i22	a32*i23	a42*i24	a12*i23	a22*i24	a32*i21	a42*i22	
				a13*i31	a23*i32	a33*i33	a43*i34	a13*i33	a23*i34	a33*i31	a43*i32	
PE3-2				a11*i12	a21*i13	a31*i14	a41*i11	a11*i14	a21*i11	a31*i12	a41*i13	
				a12*i22	a22*i23	a32*i24	a42*i21	a12*i24	a22*i21	a32*i22	a42*i23	
				a13*i32	a23*i33	a33*i34	a43*i31	a13*i34	a23*i31	a33*i32	a43*i33	
PE4-1					a11*i11	a21*i12	a31*i13	a41*i14	a11*i13	a21*i14	a31*i11	a41*i12
					a12*i21	a22*i22	a32*i23	a42*i24	a12*i23	a22*i24	a32*i21	a42*i22
					a13*i31	a23*i32	a33*i33	a43*i34	a13*i33	a23*i34	a33*i31	a43*i32
					a14*i41	a24*i42	a34*i43	a44*i44	a14*i43	a24*i44	a34*i41	a44*i42
PE4-2					a11*i12	a21*i13	a31*i14	a41*i11	a11*i14	a21*i11	a31*i12	a41*i13
					a12*i22	a22*i23	a32*i24	a42*i21	a12*i24	a22*i21	a32*i22	a42*i23
					a13*i32	a23*i23	a33*i34	a43*i31	a13*i34	a23*i31	a33*i32	a43*i33
					a14*i42	a24*i43	a34*i44	a44*i41	a14*i44	a24*i41	a34*i42	a44*i43
value					c11	c22	c33	c44	c13	c24	c31	c42
					c12	c23	c34	c41	c14	c21	c32	c43

Fig.3.7 One-D Systolic Array Architecture Refinement for Matrix Multiplication

In 4×4 matrix calculation, the cycle counts will be decreased to $N + 2N = 12$ cycles ($N = 4$), 4×4 matrix calculation will decrease 8 cycles. The 16 results of two-D 4×4 matrix needs $2 \times (N + 2N) = 24$ cycles, the latency is $(N + 2N) + N = 16$ cycles, it is scheme in Fig.3.5. Because the architecture has two 4×4 matrix multiplication blocks, when last 4×4 matrix access transpose function, next 4×4 matrix can calculate in the other matrix multiplication blocks, it can promote the throughput.

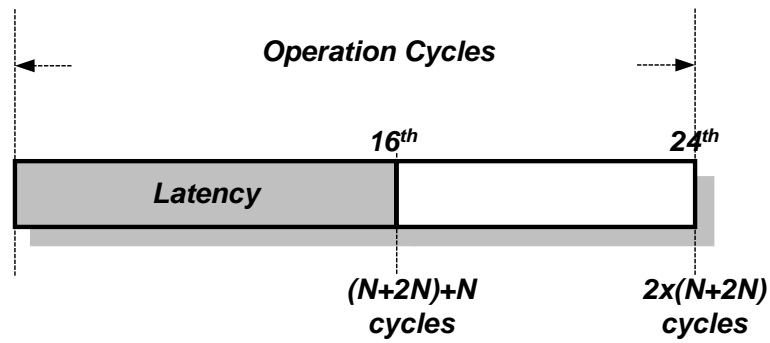


Fig.3.8 The Operation Cycles and Latency of Two-D 4x4 Matrix added more PE's

The 64 results of MPEG-2 two-D 8×8 matrix needs $2 \times \{2 \times [(N + 2N) + 1]\} = 52$ cycles, and the latency is $2 \times [(N + 2N) + 1] + [(N + 2N) + 1] = 39$ cycles and produces 32 results. The scheme is listed in Fig.3.9.

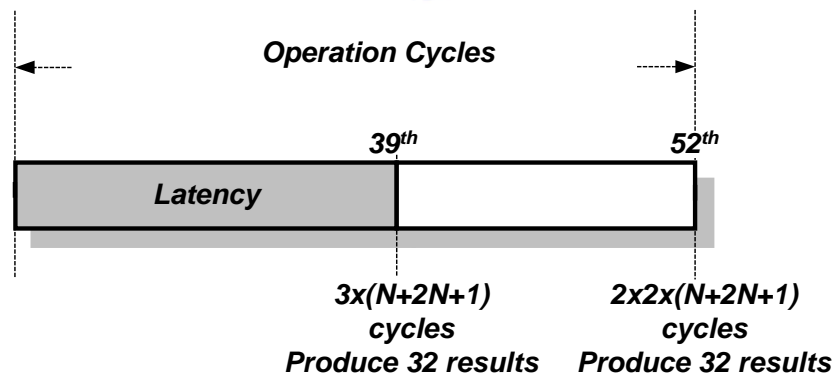
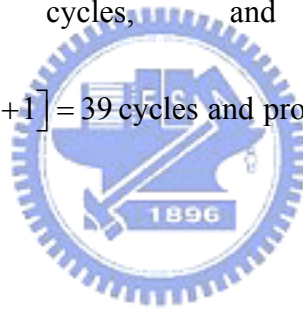



Fig.3.9 The Operation Cycles and Latency of Two-D 8x8 Matrix added more PE's

3.3 Refinement of Matrix Multiplication

The matrix multiplication of the prototype IDCT architecture uses multipliers to calculate the result. However, the multipliers take up more space on the chip and hence a method needs to be established to reduce the required chip area by eliminating the multipliers used. In section 3.3.1 canonical signed digit (CSD) and modified canonical signed digit (Modified CSD) will be introduced. In section 3.3.2, both the zero value skip and CD value skip are considered to optimize the architecture.

3.3.1 Canonical Signed Digit (CSD) and Modified

Canonical Signed Digit (Modified CSD)



The multipliers transfer two's complement and use shifters and additions based on the multiplied values or use CSD (Canonical Signed Digit) to transfer multipliers. In general, the conversion of two's complement number $B = b_{n-1}, b_{n-2}, \dots, b_0$ to the CSD from $D = d_{n-1}, d_{n-2}, \dots, d_0$ can be described in Fig.3.10. The benefit of CSD is that it optimizes the least 1's amount.

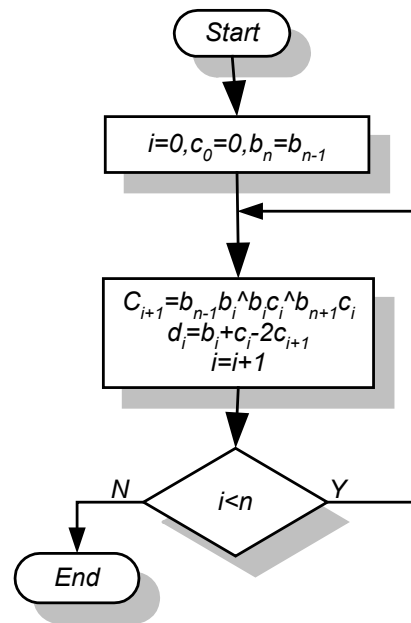


Fig.3.10 The CSD Conversion Algorithm

The standard algorithm of the conversion from the two's complement to the CSD representation does not consider the above conclusion, i.e. treats an addition and subtraction as the same cost operations. The modified CSD [12] has a modified conversion algorithm so that the conversion to the -1 (or $\bar{1}$) symbol (negative one, the subtraction) takes place only if the total number of operations (non-zero symbols) decreases.

The example of results for the two's complement, CSD and modified CSD from [12] are listed in Table 3.2. When coefficients are three and eleven, the 1's amounts of three methods are equal, but CSD method includes one addition and subtraction respectively, modified CSD does not have any subtraction. The coefficient is seven, modified CSD has one subtraction, but the number of 1's is less than two's complement. When coefficient is twenty-three, CSD includes two subtractions, but modified CSD has one subtraction, and the 1's amount is also less than two's

complement.

Table 3.2 An Example of Results For The Standard CSD And Modified CSD Conversions [12]

Coeff.	Binary (Two's complement)		CSD (Canonical Signed Digit)			MCSD (Modified CSD)		
	Value	components	Value	components		Value	components	
		addition		addition	subtraction		addition	subtraction
3	11	2	10 $\bar{1}$	1	1	11	2	0
7	111	3	100 $\bar{1}$	1	1	100 $\bar{1}$	1	1
11	1011	3	10 $\bar{1}$ 01	2	1	1011	3	0
23	10111	4	10 $\bar{1}$ 00 $\bar{1}$	1	2	1100 $\bar{1}$	2	1

In the proposed architecture, MPEG-2 and H.264 use CSD and modified CSD method, In MPEG-2, $\cos(\pi/4)$ and $\cos(\pi/16)$ include five and seven +1's respectively. In CSD, they contain three and two +1's, two -1's respectively. In modified CSD, the amount of +1's and -1's of $\cos(\pi/4)$ is same as binary (two's complement), $\cos(\pi/16)$ is less than binary (two's complement), the details are listed in Table 3.3.

Table 3.3 The Binary, CSD, and Modified CSD Values of MPEG-2 Standard

Coefficient	Binary (Two's complement)			
Value	Value (14bits)	+1's	-1's	Total 1's
$\cos(\pi/4)$	01_0110_1010_0000	5	0	5
$\cos(7\pi/16)$	00_0110_0011_1110	7	0	7

Coefficient	CSD			
Value	Value	+1's	-1's	Total 1's
$\cos(\pi/4)$	10_1 $\bar{0}$ 1 $\bar{0}$ _1010_0000	3	2	5
$\cos(7\pi/16)$	00_1 $\bar{0}$ 1 $\bar{0}$ _0100_0 $\bar{0}$ 10	2	2	4

Coefficient	MCSD (Modified CSD)			
Value	Value	+1's	-1's	Total 1's
$\cos(\pi/4)$	01_0110_1010_0000	5	0	5
$\cos(7\pi/16)$	00_0110_0100_0 $\bar{0}$ 10	3	1	4

The coefficients in H.264 standard, twelve, six and three include two +1's, respectively. In CSD, they contain one and two +1's, one -1, respectively. In modified CSD, the amount of +1 and -1's of twelve, six and three is same as binary (two's complement); the details are listed in Table 3.4.

Table 3.4 The Binary, CSD, and Modified CSD Values of H.264 Standard

Coefficient	Binary (Two's complement)			
Value	Value (14bits)	+1's	-1's	Total 1's
12	00_0000_0000_1100	2	0	2
6	00_0000_0000_0110	2	0	2
3	00_0000_0000_0011	2	0	2

Coefficient	CSD			
Value	Value	+1's	-1's	Total 1's
12	00_0000_0001_0100	1	1	2
6	00_0000_0000_1010	1	1	2
3	00_0000_0000_0101	1	1	2

Coefficient	MCSD (Modified CSD)			
Value	Value	+1's	-1's	Total 1's
12	00_0000_0000_1100	2	0	2
6	00_0000_0000_0110	2	0	2
3	00_0000_0000_0011	2	0	2

Table 3.5 lists the operators used for different types. If CSD or Modified CSD is used to do the calculation, the architecture needs approximately 108 additions/subtractions. The operators are less than using binary method.

Table 3.5 Estimate Operators for 4x4 Matrix Multiplication Architecture

	<i>Binary (Two's complement)</i>	<i>CSD</i>	<i>Modified CSD (MCSD)</i>
<i>Operators</i>	172	108	108

The new version of the architecture shows in Fig.3.11. This version does not offer the multiplied matrix to calculate matrix multiplication and replaced by CSD and modified CSD methods, the matrix multiplication calculation has been replaced by additions and subtractions.

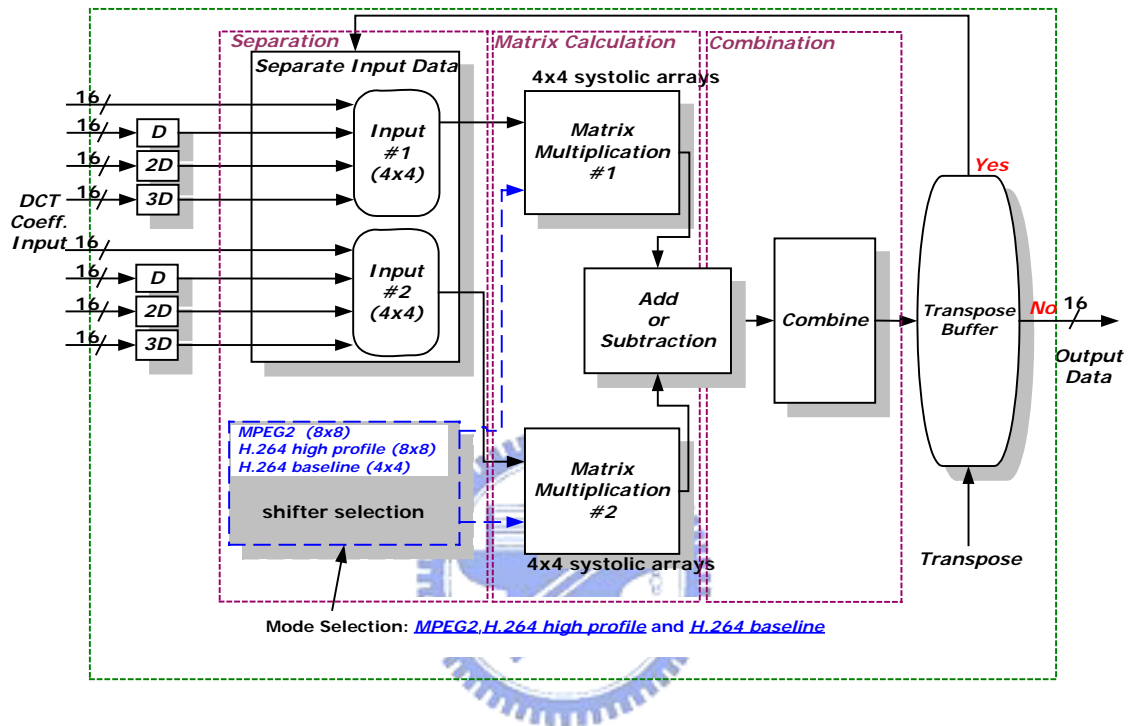


Fig.3.11 New Version IDCT Architecture Using Replaced Multiplication

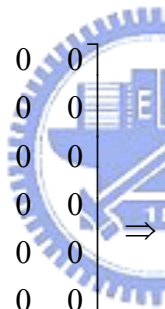
3.3.2 Zero Skip and DC Value Skip

When input value includes only DC value or all zero, do not access the multiplication process and generate the result directly.

$$\begin{bmatrix} n_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ n_j & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} n_i & n_i & n_{ii} & n_i \\ n_i & n_i & n_i & n_i \\ n_i & n_i & n_i & n_i \\ n_i & n_i & n_i & n_i \\ n_j & n_j & n_j & n_j \\ n_j & n_j & n_j & n_j \\ n_j & n_j & n_j & n_j \\ n_j & n_j & n_j & n_j \end{bmatrix}$$

Fig.3.12 DC Value Skip

As Fig.3.12 shows, if the values of the input matrix data are zero, or include DC values only as represented in Fig.3.13, matrix multiplication can be skipped.



$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig.3.13 Zero Value Skip

In Fig.3.14 shows, when zero/DC value detection signal detects all zero value or DC value, it does not access the multiplication process. Besides, it also has a synchronization problem.

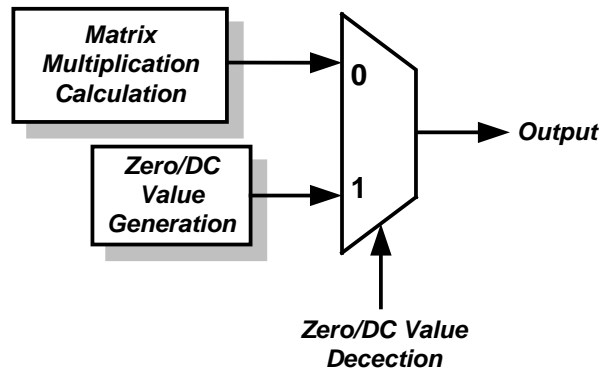


Fig.3.14 Zero/DC Value Method

The synchronization problem exists in different processing paths, our system offer a synchronizer. As Fig.3.15 shows, a Variable-Length FIFO is a synchronizer, it is for the synchronization after IDCT output interface to solve this problem.

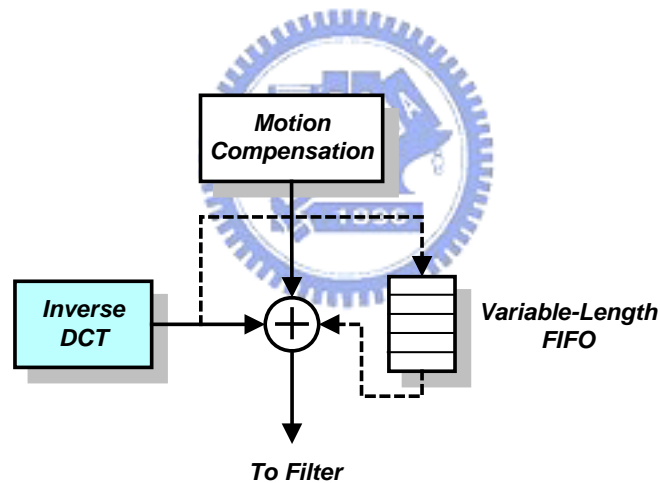


Fig.3.15 A Variable-Length FIFO for the Synchronization after IDCT Output Interface

Chapter 4

Simulation and Implementation Results

From chapter 3, we scheme one architecture includes different standards, if more standards are published in the future or many standards include different matrix type, find the different coefficients relationship and re-arrangement, one architecture can cover more standards and it can reduce the use cost, more widely used in different products. Base on one architecture, find the method to reduce the adder used and the +1's and -1's ratio. It also can reduce the required area on the chip.

In this chapter, we scheme the verification curves of simulation results and the data of implementation result. Now the architecture uses three standards (MPEG-2, H.264 high profile and H.264 baseline) to simulate the result. In section 4.1 we estimate the bit amount to describe MPEG-2 floating coefficients by table and curve analysis. In section 4.2, the implementation result will show the gate counts reduction in different methods due to architecture optimization, comparison with other related solutions will be down then.

4.1 Simulation Results

The MPEG-2 coefficient matrix is composed by cosine function, and incorporates

motion compensation and motion estimation, and so it needs to refer to the prior image in the calculation. However, if the error range is too big, the image processing will have the “Drift” problem. Therefore, IEEE has defined the accurate values [13] for peak mean square error (PMSE), peak mean error (PME), peak error (PE), overall mean square error (OMSE), and overall mean error (OME) coefficients. Table 4.1 is the block diagram of the process [13] offers 10000 patterns to test the value, and the results are listed in Table 4.1.

From Table 4.1, the two’s complement value uses 14 bits to describe the MPEG-2 floating number.

Table 4.1 Estimate Bit Amount to Describe Floating Point of MPEG-2



<i>Item</i>	<i>Standard Spec.</i>	<i>L= - 300 H= 300</i>	<i>L= - 5 H= 5</i>	<i>L= - 256 H= 255</i>
PMSE <i>(peak mean square error)</i>	<0.06 <i>(For every pixel)</i>	0.0145	0.0115	0.0176
PME <i>(peak mean error)</i>	<0.015 <i>(For every pixel)</i>	0.0023	0.0031	0.0023
PE <i>(peak error)</i>	≤1	1	1	1
OMSE <i>(overall mean square error)</i>	<0.02	0.011825	0.00905	0.01462
OME <i>(overall mean error)</i>	<0.0015	0.000134	0.000122	0.000189

Figures from 4.1 to 4.3 are the input pixel range of PMSE, PME, OMSE and OME curve diagrams.

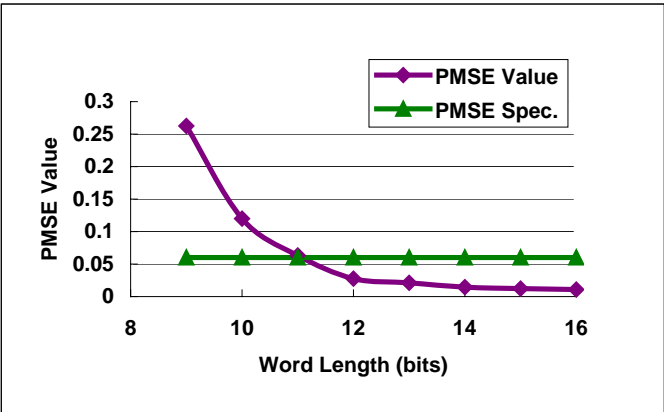


Fig.4.1 PMSE Value

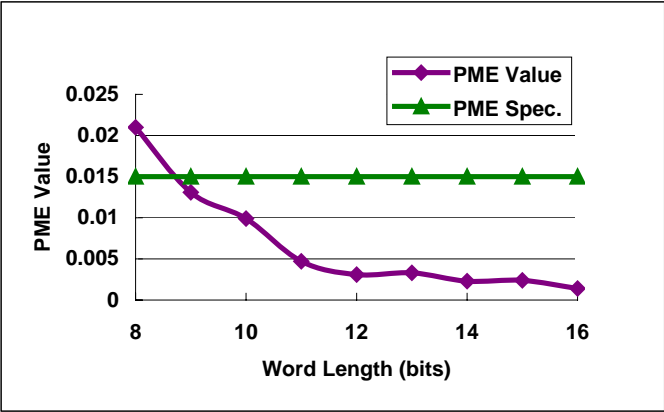


Fig.4.2 PME Value

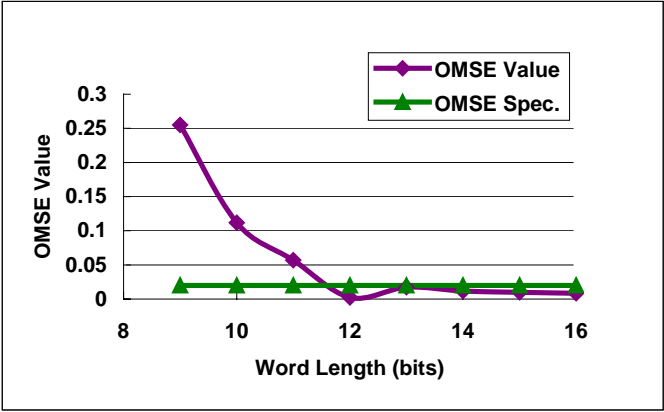


Fig.4.3 OMSE Value

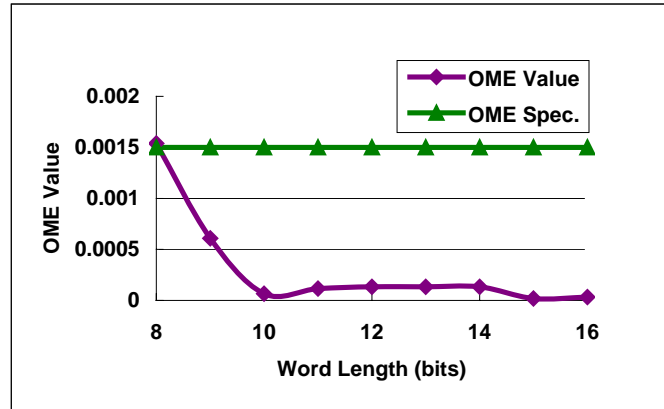


Fig.4.4 OME Value

4.2 Implementation Results

In the proposed architecture, we use some methods to optimize the matrix multiplication calculation. In section 3.3.1, the canonical signed digit (CSD) and modified canonical signed digit (Modified CSD) are described. The comparison of these methods for hardware sharing is listed in Table 4.2.

The original method uses binary (two's complement), the total gate counts are 71.73k, if we replace to CSD method, the total gate counts need becomes 46.4k, which is reduced by 35.31% in matrix calculations. On the other hand, if modified CSD is used modified CSD method is used, the total is 42.65k, it reduced 40.54% gate counts from binary and reduced 8.08% ones from CSD. The bar graph is shown in Fig.4.5.

Table 4.2 Gate Counts Reduction in Matrix Calculation

		Gate Counts (k)		
		Binary (Two's complement)	CSD (Canonical Signed Digit)	MCSD (Modified CSD)
Sub-Module (for Matrix Calculation)	Sub-Module#1	33.97	23.59	20.79
	Sub-Module#2	37.76	22.81	21.86
	Total	71.73	46.4	42.65
	% of Reducion		35.31%	40.54%
	% of Reducion			8.08%

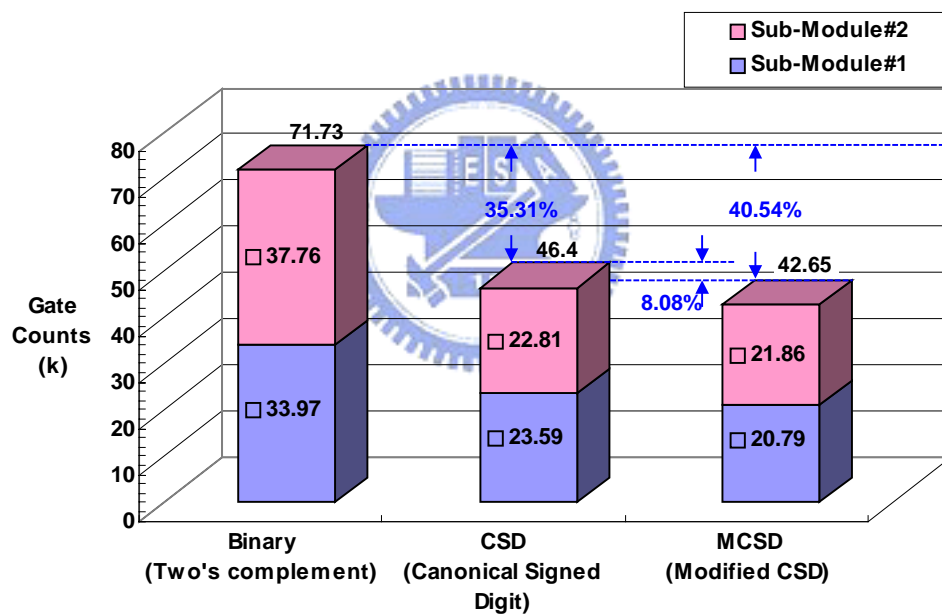


Fig.4.5 The Bar Graph of Gate Counts Reduction in Different Methods

Table 4.3 lists the gate counts of each module. Note that our proposed architecture can separate three parts, which is shown in Fig.3.6. The separation part is to separate 8 × 8 matrix into 4 × 4 of MPEG-2 and H.264 high profile. It possesses 15.91% of the overall system. The calculation part is calculate to matrix multiplication calculation, it

owns over half gate counts of overall system, is 58.59%. The combination part is to combine 4x4 matrix into 8x8 of MPEG-2 and H.264 high profile. It possesses 25.51% of the overall system. The percentage distribution for each part is shown in Fig.4.6.

Table 4.3 Gate Counts of Each Part and Comparison

<i>Parts</i>	<i>Gate Counts (k)</i>	<i>% of Overall System</i>
<i>Separation</i>	11.58	15.91%
<i>Calculation</i>	42.65	58.59%
<i>Combinaton</i>	18.57	25.51%
<i>Total</i>	72.8	100.00%

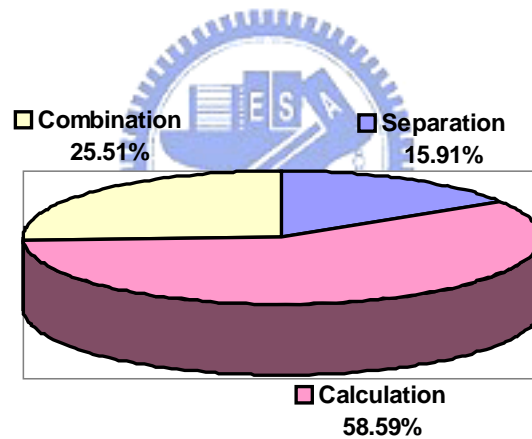


Fig.4.6 The Percentage Distribution of Gate Counts for Each Part

Table 4.4 and Table 4.5 list the comparison in state-of-art-works. Table 4.5 lists D.W Kim [14] A. Madisetti [15], and J.I Guo [16] use operators in architecture. In these existing solutions, they use two one-dimension procedures to calculate. About the operator category, the three designs use additions and the proposed architecture includes additions and subtractions by CSD and modified CSD methods.

Table 4.4 Operators List and Compare with State-of-the-Art Works

<i>Designs</i>	<i>Operators amount of 8x8 Matrix</i>	<i>Remarks</i>
<i>D.W Kim [14]</i>	144	1-D calculation Hardwired DA method, Radix-2 multibit coding
<i>A.Madisetti [15]</i>	136	1-D calculation Hardwired multiplications, Signed digit representation
<i>J.I Guo [16]</i>	132	1-D calculation Hardwired multiplications, Cyclic convolution, Signed digit representation, Common sub-expression sharing
<i>Proposed</i>	108	1-D calculation Systolic arrays, CSD/Modified CSD

Table 4.5 lists the comparison between the proposed design and some existing solutions in literatures. J.I Guo [16] is for 8x8 matrix calculation, and J D Bruguera [17] is for H.264 standard. The proposed design includes MPEG-2 and H.264 high profile 8 x8 matrix and H.264 baseline 4x4 matrix.

Table 4.5 Compare with Other Approaches

	<i>J.I Guo [16]</i>	<i>J.D Bruguera [17]</i>	<i>Proposed</i>
<i>Supporting Standards</i>	8x8 cosine coeff.	H.264	MPEG-2 H.264
<i>Technology</i>	0.35 μ m	AMS 0.35 μ m	UMC 0.18 μ m
<i>Frequency (MHz)</i>	97MHz	67MHz	125MHz
<i>Operators</i>	132	N/A	108
<i>Gate Counts</i>	62.037k	23.8k	72.8k

Based on Table 4.5, because the proposed architecture can support more standards, but about in more gate counts, a single architecture that includes multiple video

standards, the worst case scenario of each standard needs to be taken into account as different standards have different characteristics and advantages/disadvantages. Given the nature of the hardware-sharing architecture, it is not easy to fully utilize all the benefits offered by each standard. For example, different standards have their own matrix coefficients. H.264 standard has the integral coefficients, but the coefficient matrix of MPEG-2 is floating coefficients, it need more word length to describe. Consider the input data, MPEG-2 has 12bits word length for input data, but the data length of H.264 is 16bits, they need the extra overheads (e.g. needs more gate counts) in hardware sharing architecture design and implementation.

In our work, we implemented an IDCT hardware sharing architecture. Fig.4.7 shows the layout of the proposal architecture. The total gate count is about 72800 in 0.18 μ m UMC technology.

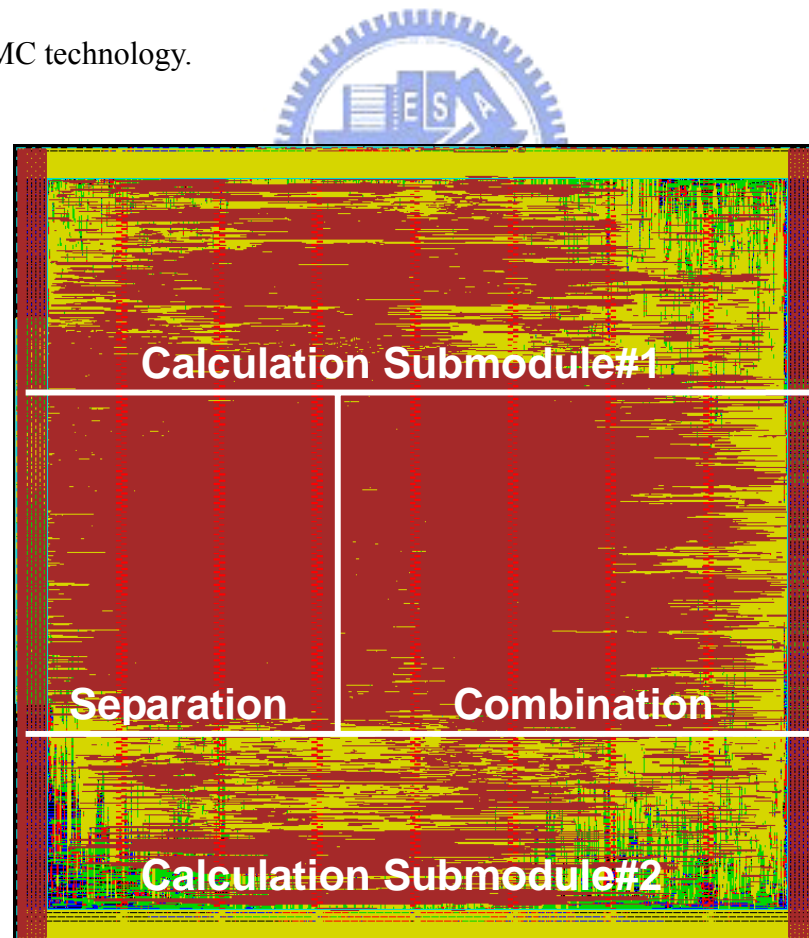


Fig.4.7 Layout of This Work

Chapter 5

Conclusions

The proposed hardware-sharing architecture for inverse discrete cosine transform (IDCT) includes three different video compression standards, namely MPEG-2, H.264/AVC high profile, and H.264/AVC baseline. With regards to reducing the required area on the chip, in matrix multiplication, the canonical signed digit (CSD) method will minimize the number of 1's, while the modified canonical signed digit (Modified CSD) method will provide the least number of -1's. As a result, a total of only 108 additions/subtractions are required to perform matrix calculation operations. The required chip area is further reduced by about 8.1%.

We have described one IDCT architectural proposal covering different standards. However many and different video coding standards will be proposed in the near future, where different matrix types (e.g. 4×4 or 8×8) and coefficients matrix will be involved. Exploration on IDCT coefficients relationship and re-arrangement will be needed to drive a more flexible solution for multi-mode multi-standard applications.

Bibliography

- [1] N. Ahmed, T. Natarajan, K. R Rao, “**Discrete Cosine Transform,**” *IEEE Transactions Computer*, vol. COM-23 pp.88-93 Jan.1974.
- [2] M. T Sun, T. C Chen A.M Gottlieb, “**VLSI implementation of a 16x16 discrete cosine transform,**” *IEEE Trans. On Circuits and Systems-II*, vol.36, no.4, pp.610-616, 1989.
- [3] D. Slwecki and W. Li, “**DCT/IDCT professor design for high data rate image coding,**” *IEEE Trans. On Circuits and Systems for Video Technology*, vol.2 no.2, pp.135-146, 1992.
- [4] T. S Chang, C. S Kung, and C. W Jen, “**A simple professor core design for DCT/IDCT,**” *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 10,No. 3 pp.439-447, April 2000.
- [5] P. N Tudor “**Tutorial MPEG-2 Video Compression,**”*Electric and Communication Engineering Journal* Dec. 1995 pp. 257–264.
- [6] ISO/MPEG “**Generic Coding of Moving Pictures and Associated Audio Committee Draft,**”Nov. 1993.
- [7] W. H Chen, C.H Smith S.C Fralick, “**A Fast computatuinal algorithm for the Discrete Cosine Transform,**” *IEEE Trans. On Communications* vol.COM-25, vol. 9, pp.1004-1009, Sep. 1977.
- [8] B. G Lee “**A new Algorithm to compute the Discrete Cosine Transform,**” *IEEE Transactions on ASSP*, Vol. ASSP-32,No.6, pp.1243-1245, Dec.1984.
- [9] N. I Cho, S.U Lee “**DCT algorithms for VLSI parallel implementation,**” *IEEE Trans. On Acoustics, Speech, and Signal Processing*, col.38, no.1, pp.121-127,

1990.

- [10] M. A Bayoumi, G. A Jullien, and W .C Miller “**A VLSI Array for Computing the DFT Based on RNS,**” in *Proc. ICASSP 86' Tokyo*, pp. 2147-2150.
- [11] T. Komarek, P. Pirsch “**Array architectures for block matching algorithms,**” *IEEE Transactions On Circuits And Systems*, Vol. 36, No. 10, October 1989.
- [12] K. Wiatr, E. Jamro “**Constant Coefficient Multiplication in FPGA Structures,**” *Euromicro Conference, 2000. Proceedings of the 26th* Vol. 1,5-7 Sept. 2000 pp. 252-259.
- [13] “**IEEE Standard Specifications for the Implementation of 8x8 Inverse Discrete Cosine Transform,**” *IEEE standard*, 1180-1990, Mar 1991.
- [14] D.W Kim, et Al., “**A Compatible DCT/IDCT Architecture Using Hardwired Distributed Arithmetic,**” *ISCAS Proc.*,pp.II457-II460, 2001.
- [15] A. Madiseti and A.N Willson Jr., “**A 100MHz 2-D 8×8 DCT/IDCT Processor for HDTV Applications,**” *IEEE Trans. on circuits and systems for video technology*, vol.2, no.2, pp.135-146, 1995.
- [16] J. I Guo, R. C Ju, and J. W Chen, “**An Efficient 2-D DCT/IDCT Core Design Using Cyclic Convolution and Adder-Based Realization,**” *IEEE Trans.on circuits and systems for video technology*, under revision, 2003.
- [17] J. D Bruguera and R. R Osorio, “**A Unified Architecture for H.264 Multiple Block-Size DCT with Fast and Low Cost Quantization,**” *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th Euromicro Conference on*, pp.407-414, Aug 2006.

About the Author

姓 名：洪岳琪 Yueh-Chi Hung

出 生 地：台灣省雲林縣

出生日期：1974. 12. 22

學 歷：

1981. 9 ~ 1987. 6 台北市立士林國民小學

1987. 9 ~ 1990. 6 台北市立士林國民中學

1990. 9 ~ 1995. 6 新埔工業專科學校 電機工程科

1995. 9 ~ 1998. 6 中國文化大學 電機工程學系 學士

2005. 2 ~ 2007. 1 國立交通大學 電機學院

IC 設計產業研發碩士班 碩士

經 歷：

2000. 7 ~ 2001. 2 台朔光電股份有限公司 實習工程師

2001. 3 ~ 2004. 6 明基電通股份有限公司 影存事業群

SPC 事業部 研發一部 工程師

發表專利：

- 洪岳琪，應用預覽掃描來提昇掃描影像品質的方法與裝置。

Method and Apparatus for Improving Quality of a Scanned Image Through a Preview Operation.

申請號碼：93107271，申請日：2004/03/18

- 洪岳琪、李俊仁，影像處理裝置及其控制馬達系統之方法。

Image Processing Device and Method for Controlling a Motor System.

申請號碼：93107010，申請日：2004/03/16

得 獎 事 績



2006. 5 94 學年度積體電路設計競賽

標準單元設計 佳作獎

發 表 論 文

- Yi-Hong Huang, Ping-Chang Lin, Kang-Cheng Hou, Yueh-Chi Hung, Tsu-Ming Liu, Chen-Yi Lee,” A High-Throughput SRAM-Based Context Adaptive Binary Arithmetic Decoder (CABAD) for H.264/AVC”, in Proceedings of the 17th VLSI/CAD Symposium, August 2006.