

# HARDWARE EFFICIENT DESIGN OF PARALLEL FIR FILTERS AND ITS APPLICATION TO 2D DWT

## A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

**MASTER OF TECHNOLOGY**

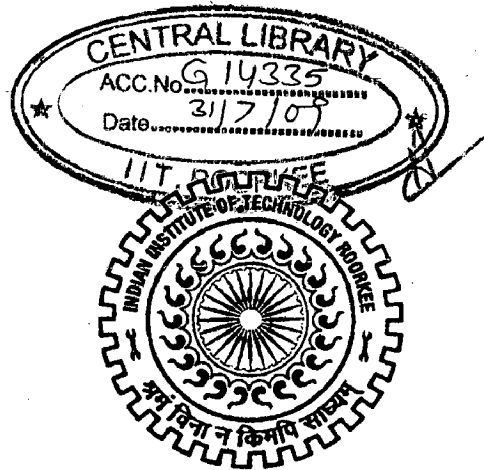
*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**(With Specialization in Semiconductor Devices & VLSI Technology)**

By

**SURESH KELLAMPALLI**



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY ROORKEE**

**ROORKEE -247 667 (INDIA)**

**JUNE, 2008**

## Candidate's Declaration

I hereby declare that the work being presented in the dissertation report titled "HARDWARE EFFICIENT DESIGN OF PARALLEL FIR FILTERS AND ITS APPLICATION TO 2D DWT" in partial fulfillment of the requirement for the award of the degree of Master of Technology in Semiconductor Devices & VLSI Technology, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr. S.DASGUPTA, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.

I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 26/06/08

  
(SURESH KELLAMPALLI)

Place: IIT Roorkee.


---

## Certificate

This is to certify that above statements made by the candidate are correct to the best of my knowledge and belief.

Dated: 26.06.2008

Place: IIT Roorkee.

  
Dr.S.DASGUPTA

Assistant Professor,  
Department of Electronics and  
Computer Engineering, IIT Roorkee,  
Roorkee -247667 (India).

## **ACKNOWLEDGEMENTS**

I am thankful to Indian Institute of Technology Roorkee for giving me this opportunity. It is my privilege to express thanks and my profound gratitude to my supervisor Dr.S.Dasgupta, Assistant Professor for his invaluable guidance and constant encouragement throughout the dissertation.

I am also grateful to the staff of VLSI Design Laboratory for their kind cooperation extended by them in the execution of this dissertation. I am thankful to research scholars of VLSI, who helped me consistently in doing this work. I am also thankful to all my friends who helped me directly and indirectly in completing this dissertation.

Most importantly, I would like to extend my deepest appreciation to my family for their love, encouragement and moral support. Finally I thank God for being kind to me and driving me through this journey.

**(SURESH KELLAMPALLI)**

## ABSTRACT

Finite impulse response (FIR) filter is regarded as one of the major operations in digital signal processing. Parallel processing is a powerful technique because it can be used to increase the throughput of a FIR filter or reduce the power consumption of a FIR filter. Parallel FIR filters (i.e., realizing FIR filters in parallel) has got its various applications in 2D Discrete wavelet Transform (DWT), Motion Estimation in Video Compression, Equalizers and 2D FIR filters. However, a traditional parallel FIR filter implementation causes a linear increase in the hardware cost (area) by a factor of  $L$ , the block size i.e., level of parallelism. In many design situations, this large hardware penalty cannot be tolerated. Therefore, it is advantageous to produce parallel FIR filter implementations that require less area than traditional parallel FIR filtering structures.

An approach to increase the throughput of FIR filters, with reduced complexity hardware based on fast FIR algorithms and fast short length linear convolution algorithms, were presented. Although their basic idea is the same, i.e., first derive smaller length fast parallel filters and then cascade or iterate these short-length filters for long block sizes, their starting point is not the same. These methods will have a simple and efficient control in the increase of hardware cost. These hardware efficient parallel FIR filters can be used for the fast implementation of 2D Discrete Wavelet Transform (DWT) than the other convolution and lifting based architectures.

The focus of this thesis is to present the recent methods to realize the hardware efficient parallel FIR filters (i.e., by Fast FIR algorithms and Fast short convolution algorithms) and its application to 2D Discrete Wavelet Transform. The hardware simulation of these efficient structures are carried out in Modelsim and synthesized using Xilinx. A Matlab code is developed for finding the computational complexity of each method. The comparisons of these methods are also done. For 2D DWT the comparisons are done with the recent convolution based architectures.

# TABLE OF CONTENTS

ABSTRACT .....	iii
LIST OF FIGURES .....	vi
LIST OF TABLES .....	viii
<b>1: Introduction to Parallel FIR filters.....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Organization of Thesis .....	3
<b>2: Formulation of Parallel FIR filters using Polyphase Decomposition .....</b>	<b>4</b>
<b>3: Parallel FIR filters based on fast FIR Algorithms .....</b>	<b>8</b>
3.1 Fast FIR Algorithms .....	8
3.1.1 Two parallel Fast FIR filter.....	8
3.1.2 Matrix representation .....	9
3.1.3 Parallel filters by transposition.....	10
3.1.4 Three parallel fast FIR filter.....	11
3.1.5 Fast Parallel FIR algorithm for large block sizes .....	13
3.2 Fast FIR filters based on Frequency Spectrum.....	15
3.2.1 FFA structures for 2-parallel and 3-parallel FIR filters .....	16
3.3.1 Cascading FFA's .....	18
3.3.2 Selection of FFA types .....	20
<b>4: Parallel FIR Filter structures based on fast convolution Algorithms.....</b>	<b>21</b>
4.1 Fast Parallel FIR filters based on linear convolution .....	21
4.2 Iterated Short Convolution Algorithm(ISCA) .....	22
4.2.1 Complexity Computation .....	26
4.3 Improved Fast Parallel FIR filter structure .....	27
4.3.1 Improved structure Algorithm .....	28
4.3.2 Complexity Computation .....	30
<b>5: Parallel FIR filters based on 2-stage parallelism .....</b>	<b>32</b>
5.1 2-stage Parallelism.....	32
5.1.1 Generalization (Method-1) of 2-stage Parallelism .....	32

5.1.2 Method-2 of 2-stage parallelism .....	33
5.1.3 Complexity Computation .....	33
5.1.4 Example for 2-Stage Parallelism realization .....	35
5.1.4.1 6-Parallel FIR Subfilter as a Shared Filtering Core .....	40
5.1.4.2 Method-2 realization .....	41
<b>6: Application of Parallel FIR filters .....</b>	<b>43</b>
6.1 2D Discrete Wavelet Transform .....	43
6.1.1 2D Non separable DWT structure based on parallel FIR filters.....	43
6.1.2 2D DWT of an Image by L- Parallel FIR Filtering .....	50
6.1.3 Complexity Computation .....	53
<b>7: Results and Discussion.....</b>	<b>55</b>
7.1 Hardware Simulation.....	55
7.1.1 Simulation results of FFA structures .....	55
7.2.2 Simulation results of ISCA structures .....	58
7.1.3 Simulation results of 2-stage parallelism .....	59
7.1.4 Simulation results of 2D DWT based on parallel FIR filters.....	61
7.2 Comparison and Analysis.....	62
7.2.1 FFA VS ISCA .....	62
7.2.2 ISCA Vs 2-stage Parallelism .....	64
7.2.3 Complexity for 2D DWT .....	67
<b>8: Conclusion .....</b>	<b>68</b>
<b>REFERENCES.....</b>	<b>69</b>
<b>APPENDIX A :COOK-TOOM Algorithm .....</b>	<b>72</b>
<b>APPENDIX B :WINOGRAD Algorithm.....</b>	<b>75</b>
<b>APPENDIX C :Some Efficient Linear Convolutions .....</b>	<b>78</b>

## LIST OF FIGURES

1.1 Transposed form FIR filter .....	01
1.2 Sequential and 3-parallel architecture of an FIR filter .....	02
2.1 Traditional 2-parallel FIR filter .....	05
2.2 Traditional 3-parallel FIR filter .....	06
3.1 Reduced complexity 2-parallel fast FIR implementation .....	09
3.2 Transposed reduced complexity 2-parallel FIR filter .....	11
3.3 Reduced complexity 3-parallel FIR implementation.....	13
3.4 4-parallel fast FIR filter by cascading two 2-parallel FFA's( $F_0$ and $F_1$ ) .....	15
3.5 FFA1 structure of 2-parallel fast FIR filter.....	16
3.6 FFA1 structure for 3-parallel FIR filter .....	17
3.7 FFA2 structure for 3-parallel FIR filter .....	18
3.8 FFA0 or FFA1 4-parallel FIR filter structure .....	19
3.9 FFA1' structure of 4-parallel FIR filter .....	19
4.1 ISCA based 4-parallel FIR filter.....	26
4.2 Implementation of N-tap FIR filter with $N \times N$ linear convolution.....	28
4.3 2-parallel 6-tap FIR filter.....	29
4.4 An improved 3-parallel structure for 2-parallel 6-tap FIR filter.....	29
4.5 (a) $3 \times 3$ Delay Element Matrix (DEM) (b) Delay element function .....	30
4.6 Shape of Delay Element Matrix (a) $\prod_{i=1}^r M_i > N/L$ (b) $\prod_{i=1}^r M_i < N/L$ .....	31
5.1 Implementation of 3-parallel FIR filter .....	35
5.2 2-stage(method-1) parallelism of 3-parallel 36-tap FIR filter .....	36
5.3 (a) $6 \times 6$ DEM (b) Delay element function.....	37
5.4 Preloading $6 \times 6$ DEM when (a) $k = 0$ and $i = 0$ (i.e., $t = 0$ ), (b) $k = 0$ and $i = 5$ (i.e., $t = 5$ ).....	38
5.5 Preloading $6 \times 6$ DEM when (a) $k = 6$ and $i = 0$ (i.e., $t = 6$ ), (b) $k = 6$ and $i = 5$ (i.e., $t = 11$ ).....	38
5.6 Postloading $6 \times 6$ DEM when (a) $k = 12$ and $i = 0$ (i.e., $t = 0$ ), (b) $k = 12$ and $i = 5$ (i.e., $t = 17$ ).....	38

5.7 Postloading $6 \times 6$ DEM when (a) $k = 17$ and $i = 0$ (i.e., $t = 0$ ), (b) $k = 17$ and $i = 5$ (i.e., $t = 23$ ) .....	39
5.8 Timing of the 3-parallel 36-tap filter by 2-stage parallelism.....	39
5.9 (a) 6-parallel FIR filter as shared filtering core (b) block diagram of (a).....	40
5.10 2-stage parallel FIR filter for an 6-parallel 36 tap FIR filter.....	42
5.11 Timing of 6-parallel 36-tap FIR filter.....	42
6.1 2-parallel 2-tap FIR filter .....	45
6.2 1D DWT (i.e., after row filtering) based on 2-parallel FIR filter .....	46
6.3 Input data flow of 2D DWT of 2-parallel FIR filter .....	48
6.4 Output data flow of 1D DWT of 2-parallel FIR filter .....	48
6.5 (a) $N^2/4$ 2D DWT structure for an $8 \times 8$ image (b) $4 \times 4$ Delay Element Matrix (c) Delay Element function.....	50
6.6 Output data flow of $4 \times 4$ DEM.....	50
6.7 $N^2/2L$ 2D DWT structure for an $N \times N$ image.....	51
6.8 Output data flow of the first level $N^2/4$ 2D DWT structure.....	51
6.9 Interleaving structures of 2D DWT structures for an $8 \times 8$ image.....	51
6.10 Hardware implementation of 2D DWT for an image of $8 \times 8$ size with J-level resolution in $N^2/3$ clock cycles.....	51
7.1 FFA0 2-parallel FIR filter.....	55
7.2 FFA0 3-parallel FIR filter .....	55
7.3 FFA2 3-parallel FIR filter.....	56
7.4 FFA0 4-parallel FIR filter.....	56
7.5 FFA1' 4-parallel FIR filter .....	57
7.6 FFA1 2-parallel FIR filter.....	57
7.7 FFA1 3-parallel FIR filter .....	57
7.8 FFA1 4-parallel FIR filter.....	58
7.9 ISCA based 4-parallel FIR filter.....	58
7.10 ISCA based 6-parallel 24 tap parallel FIR filter .....	59
7.11.1 Schematic of 6-parallel 36-tap FIR filter .....	59
7.11.2 2-stage 6-parallel 36-tap FIR filter .....	60



7.12 2-stage 3-parallel 36-tap FIR filter .....	60
7.13 Schematic of 2D DWT for an $8 \times 8$ image based on 2-parallel 2-tap FIR filter ....	61
7.14 Non separable 2D DWT for an $8 \times 8$ image based on 4-parallel 2-tap FIR filter...	61
7.15 Hardware cost (i.e., complexity) comparison between ISCA and FFA for $N=144,576,1152$ at different levels of parallelism.....	64
7.16 Comparison of complexities of $N=144,1152$ between ISCA and 2-stage parallelism at different levels of parallelism .....	66

## LIST OF TABLES

6.1 Data flow of 2-D DWT of a $8 \times 8$ image with 2-level resolution and computation of $N^2/3$ clock cycles.....	54
7.1 Comparison between 2D DWT structure and previous convolution based architectures for an $N \times N$ image.....	67

# 1. Introduction to Parallel FIR filters

## 1.1 Introduction

Finite impulse response (FIR) filter is regarded as one of the major operations in digital signal processing. A linear time invariant (LTI) FIR filter [24] is one of the basic building blocks common to most DSP systems. The output of an FIR filter is a sequence generated by convolving the sequence of the input samples with  $N$  filter coefficients. The filter expression can be described by

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \quad (1.1)$$

where  $N$  is the length of the filter (i.e.  $N-1$  is the order),  $h(k)$  denotes the  $k^{\text{th}}$  coefficient, and  $x(n-k)$  denotes the sampled input data at time  $n-k$ .

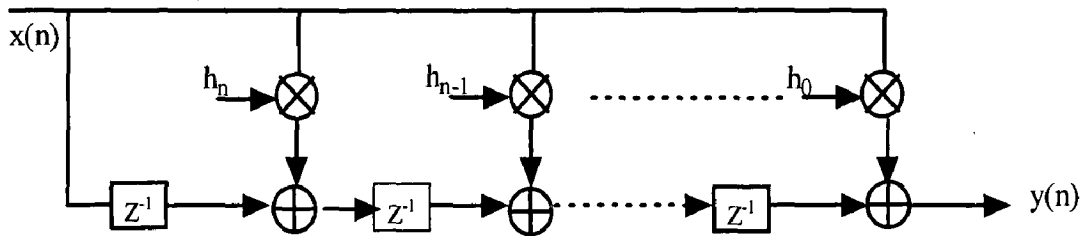


Fig 1.1. Transposed form FIR filter

The main operation of an FIR filter is convolution, which can be performed using addition and multiplication. The high computational complexity of such an operation makes the use of special hardware more suitable for enhancing the computational performance. This special hardware used to realize a high-tap-number FIR filter is costly. Thus minimizing the hardware cost of this special hardware is an important issue

Consider the 3-tap FIR filter as described in equation (1.2) i.e., from (1.1) with length  $N=3$ . The system is single input and single output (SISO) system as described by the equation

$$y(k) = ax(k) + bx(k-1) + c(k-2) \quad (1.2)$$

To obtain a parallel processing structure, the SISO system must be converted into an MIMO (multiple input and multiple output) system. For example the following set of

equations describe a parallel system with 3 inputs per clock cycle (i.e.) level of parallel processing  $L=3$ ).

$$\begin{aligned}
 y(3k) &= ax(3k) + bx(3k-1) + c(3k-2) \\
 y(3k+1) &= ax(3k+1) + bx(3k) + c(3k-1) \\
 y(3k+2) &= ax(3k+2) + bx(3k+1) + c(3k)
 \end{aligned}
 \tag{1.3}$$

Parallel processing system is also called block processing, and the number of inputs processed in a clock cycle is referred to as the block size. Each delay element is referred to as a block delay, (i.e.) delaying the signal  $x(3k)$  by 1 clock cycle would result in  $x(3k-3)$  instead of  $x(3k-1)$ , which has been input in another input line. The sequential and 3-parallel architecture for an FIR filter is shown in figure 1.2.

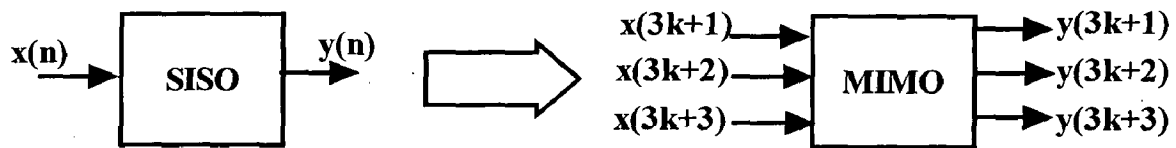


Fig 1.2 Sequential and 3-parallel architecture of an FIR filter

The critical path of the block (or parallel) processing system remains unchanged. Parallel, or block, processing can be applied to digital FIR filters to either increase the effective throughput or reduce the power consumption of the original filter. Traditionally, the application of parallel processing to an FIR filter involves the replication of the hardware units that exist in the original filter. If the area required by the original circuit is  $A$ , then the  $L$ -parallel circuit requires an area of  $L \times A$ . With the continuing trend to reduce chip size and integrate multi-chip solutions into a single chip solution, it is important to limit the silicon area required to implement a parallel FIR digital filter. In many design situations, the hardware overhead that is incurred by parallel processing cannot be tolerated due to limitations in design area. Therefore, it is advantageous to realize parallel FIR filtering structures that consume less area than traditional parallel FIR filtering structures.

## 1.2 Organization of this thesis

This thesis is organized as follows. Chapter 2 begins with parallel FIR filters based on polyphase decomposition. Chapter 3 analyzes the parallel FIR filters using Fast FIR algorithms. This is followed by the description of those filters in matrix form and its transposition. It also discusses the parallel FIR filters based on frequency spectrum and ~~also on linear convolution~~. And the Chapter 4 presents the efficient short length linear convolution algorithms by Winograd and Cook-Toom Algorithms. Then these short efficient convolutions are iteratively convolved to obtain higher efficient convolutions by Iterated Short Convolution Algorithm(ISCA). An improved structure for further reduction of hardware cost of parallel FIR filters is also discussed .Chapter 5 analyzes the two methods of 2-stage Parallelism with an example of 3-parallel 36-tap FIR filter. Chapter 6 begin with the presentation of the applications of parallel FIR filters <sup>to</sup> the high speed implementation of 2D DWT structures. Chapter 7 will show the hardware simulation results of all the structures for parallel FIR filters analyzed in previous chapters and the comparisons of one over the other are also presented. Chapter 8 concludes this thesis with suggestions for future work.

## 2 Formulation of Parallel FIR filters using Polyphase Decomposition

An  $N$ -tap FIR filter obtained from an input sequence  $x(n)$  of infinite length sequence and the impulse sequence  $h(n)$  of length  $N$ , in  $z$ -domain as

$$Y(z) = H(z)X(z) = \sum_{n=0}^{N-1} h(n)z^{-n} \sum_{n=0}^{\infty} x(n)z^{-n} \quad (2.1)$$

The input sequence  $x(n)$  can be decomposed into even-numbered part and odd-numbered part as follows

$$\begin{aligned} X(z) &= x(0) + z^{-1}x(1) + z^{-2}x(2) + z^{-3}x(3) + \dots \\ &= x(0) + z^{-2}x(2) + \dots + z^{-1}(x(1) + z^{-2}x(3) + \dots) \\ &= X_0(z^2) + z^{-1}X_1(z^2) \end{aligned} \quad (2.2)$$

Where  $X_0(z^2)$  and  $X_1(z^2)$  are  $z$ -transforms of  $x(2k)$  and  $x(2k+1)$  ( $0 \leq k \leq \infty$ ). Similarly, the length- $N$  filter coefficients  $H(z)$  can be decomposed as

$$H(z) = H_0(z^2) + z^{-1}H_1(z^2) \quad (2.3)$$

Where  $H_0(z^2)$  and  $H_1(z^2)$  are of length  $N/2$  and are referred to as even subfilter and odd subfilter. The even-numbered output sequence  $y(2k)$  and the odd-numbered output sequence  $y(2k+1)$  ( $0 \leq k \leq \infty$ ) can be computed as

$$\begin{aligned} Y(z) &= Y_0(z^2) + z^{-1}Y_1(z^2) \\ &= (X_0(z^2) + z^{-1}X_1(z^2))(H_0(z^2) + z^{-1}H_1(z^2)) \\ &= X_0(z^2)H_0(z^2) + z^{-1}(X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2)) + z^{-2}X_1(z^2)H_1(z^2) \end{aligned} \quad (2.4)$$

Where  $Y_0(z^2)$  and  $Y_1(z^2)$  correspond to  $y(2k)$  and  $y(2k+1)$  in time domain, respectively. The filtering operation in equation (2.4) processes two inputs  $x(2k)$  and  $x(2k+1)$  and generates two outputs  $y(2k)$  and  $y(2k+1)$  every iteration, and is referred to as 2-parallel FIR filter. This 2-parallel FIR filter can be rewritten in matrix form as

$$\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} H_0 & z^{-2}H_1 \\ H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} \quad (2.5)$$

Figure 2.1 shows the resulting traditional i.e. by polyphase decomposition, 2-parallel FIR filtering structure, which requires  $2N$  multiplications and  $2(N-1)$  additions.

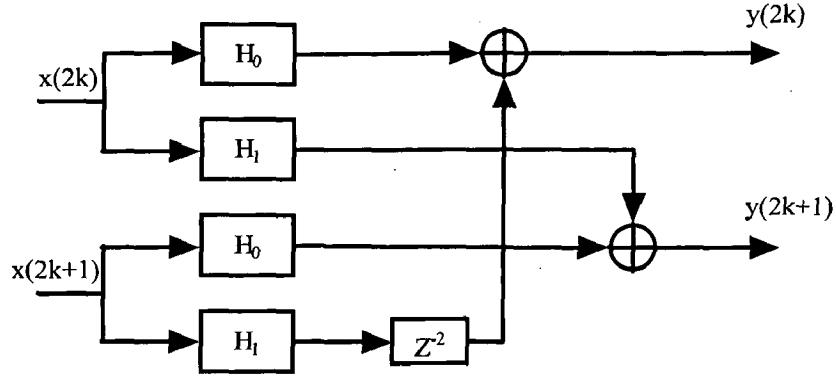


Fig 2.1 Traditional 2-parallel FIR filter

Similarly 3-phase polyphase decomposition, the input sequence  $X(Z)$  and the filter coefficients  $H(Z)$  can be decomposed as follows

$$\begin{aligned} X(z) &= X_0(z^3) + z^{-1}X_1(z^3) + z^{-2}X_2(z^3) \\ H(z) &= H_0(z^3) + z^{-1}H_1(z^3) + z^{-2}H_2(z^3) \end{aligned} \quad (2.6)$$

Where  $X_0(z^3)$ ,  $X_1(z^3)$  and  $X_2(z^3)$  correspond to  $x(3k)$ ,  $x(3k+1)$  and  $x(3k+2)$  in time domain respectively and  $H_0(Z)$ ,  $H_1(Z)$  and  $H_2(Z)$  are three sub filters of  $H(Z)$ . The output is given as

$$Y(z) = Y_0(z^3) + z^{-1}Y_1(z^3) + z^{-2}Y_2(z^3) \quad (2.7)$$

Hence,

$$\begin{aligned} Y_0(z^3) &= X_0(z^3)H_0(z^3) + z^{-3}X_1(z^3)H_2(z^3) + z^{-3}X_2(z^3)H_1(z^3) \\ Y_1(z^3) &= X_0(z^3)H_1(z^3) + X_1(z^3)H_0(z^3) + z^{-3}X_2(z^3)H_2(z^3) \\ Y_2(z^3) &= X_0(z^3)H_2(z^3) + X_1(z^3)H_1(z^3) + X_2(z^3)H_0(z^3) \end{aligned} \quad (2.8)$$

Where  $Y_0(z^3)$ ,  $Y_1(z^3)$  and  $Y_2(z^3)$  correspond to  $y(3k)$ ,  $y(3k+1)$  and  $y(3k+2)$  respectively. This 3-parallel FIR filter processes 3 input samples  $x(3k)$ ,  $x(3k+1)$  and  $x(3k+2)$  and generates 3 output samples  $y(3k)$ ,  $y(3k+1)$  and  $y(3k+2)$  in one iteration and can be rewritten in matrix form as

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} H_0 & z^{-3}H_2 & z^{-3}H_1 \\ H_1 & H_0 & z^{-3}H_2 \\ H_2 & H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} \quad (2.9)$$

Figure 2.2 shows the resulting 3-parallel FIR filtering structure, which requires  $3N$  multiplications and  $3(N-1)$  additions.

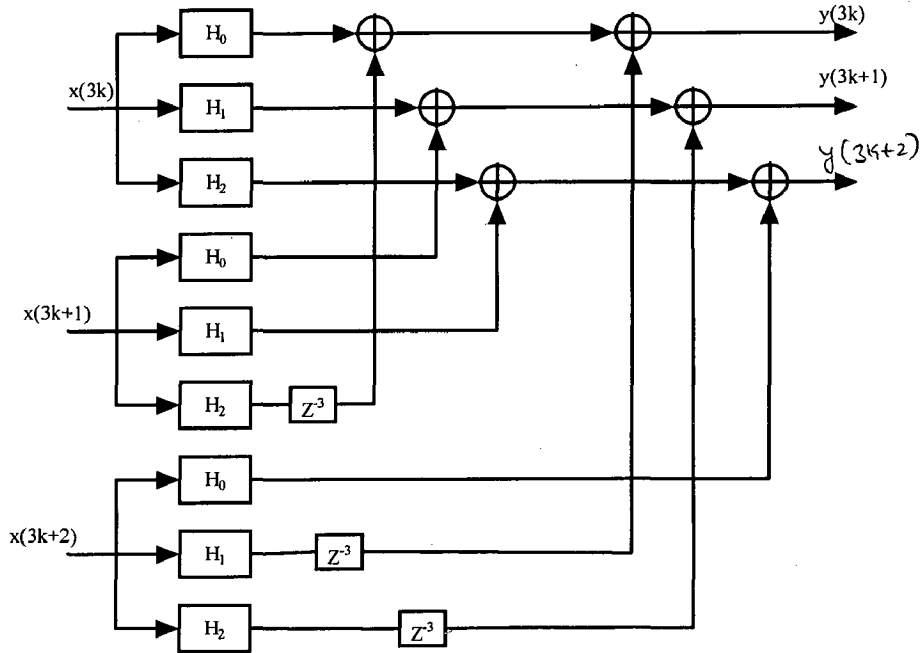


Fig 2.2 Traditional 3-parallel FIR filter

Generally, the polyphase decomposition can be used to derive  $L$ -parallel FIR filters by decomposing  $X(z)$ ,  $H(z)$  and  $Y(z)$  into  $L$  subsequences as follows:

$$\begin{aligned} X_i(z) &= \sum_{k=0}^{\infty} z^{-k} x(Lk+i), \quad i=0,1,\dots,L-1 \\ H_i(z) &= \sum_{k=0}^{(N/L)-1} z^{-k} h(Lk+j), \quad j=0,1,\dots,L-1 \\ Y_l(z) &= \sum_{k=0}^{\infty} z^{-k} y(Lk+l), \quad l=0,1,\dots,L-1 \end{aligned} \quad (2.10)$$

The  $L$  output subsequences  $y(Lk+l)$  ( $0 \leq l \leq L-1, 0 \leq k \leq \infty$ ) can be computed using an combination of  $L$  sub filters from the  $L$  input subsequences  $x(Lk+i)$  ( $0 \leq i \leq L-1, 0 \leq k \leq \infty$ ) as

$$Y_k = z^{-L} \sum_{i=k+1}^{L-1} H_i X_{L+k-i} + \sum_{i=0}^k H_i X_{k-i}, \quad 0 \leq k \leq L-2$$

$$Y_{L-1} = \sum_{i=0}^{L-1} H_i X_{L-1-i}$$
(2.11)

This can also be rewritten in matrix form as

$$Y=HX$$

$$\begin{bmatrix} Y_0 \\ Y_1 \\ \dots \\ Y_{L-1} \end{bmatrix} = \begin{bmatrix} H_0 & z^{-L}H_{L-1} & \dots & z^{-L}H_1 \\ H_1 & H_0 & \dots & z^{-L}H_2 \\ \dots & \dots & \dots & \dots \\ H_{L-1} & H_{L-2} & \dots & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ \dots \\ X_{L-1} \end{bmatrix}$$
(2.12)

It should be noted that H is a *pseudo-circulant* matrix[1] as in (2.12). This L-parallel FIR filter requires  $L^2$  sub filtering operations, each of which is of length  $(N/L)$  and requires  $(N/L)$  multiply-add operations. Hence the  $L$  parallel FIR filter requires  $L^2(N/L)$  or  $LN$  multiply-add operations, which is linear in the block size  $L$ . Although the polyphase formulation does not reduce the parallel filter complexity, it can be exploited to derive fast parallel FIR filter structures.



## 3 Parallel FIR filters based on Fast FIR Algorithms

### 3.1 Fast FIR Algorithms

Since the complexity of a traditional block filter increases linearly with the block size or the number of samples processed in parallel in a clock cycle, fast FIR algorithms are developed [2][6] to reduce the hardware complexity. This Fast Filtering Algorithms (FFA's) will reduce the complexity of parallel filtering structures. Since the work of Winograd [1], it is known that two polynomials of degree  $(L-1)$  can be multiplied using only  $(2L-1)$  product terms. Therefore, the  $L$ -parallel filter can be implemented using approximately  $(2L-1)$  filtering operations of length  $(N/L)$ . The resulting parallel filtering structure would require  $(2N - N/L)$  multiplications. For large values of  $N$ , the FFA's can reduce the number of multiplications significantly at the expense of increasing the number of additions. Replacing multipliers with add operations is advantageous because adders have a smaller implementation cost than multiplier in terms of silicon area. For large values of  $L$ , however, the number of adders becomes unmanageable. Therefore a balance between multipliers and adders can be maintained.

In the general case, a  $(n \times n)$  FFA produces a FIR filtering structure that is the functional equivalent of a parallel FIR filter of block size  $n$ . The application of a  $(n \times n)$  FFA produces a set of filters each of which are length  $N/n$ , where  $N$  is the length of the original FIR filter. The set of filters that are produced by a  $(n \times n)$  FFA will consist of the  $n$  filters,  $H_0, H_1, \dots, H_L$ , that are produced by taking the polyphase decomposition of the original filter with decomposition factor  $n$ , plus the filters that result from taking the additive combinations of these  $n$  filters. The proper filter transfer function is realized with the addition of some pre- and post-processing steps that are performed in conjunction with the filtering operations.

#### 3.1.1 Two parallel Fast FIR filter

The equations of polyphase decomposition of 2-parallel FIR filter can be rewritten as

$$\begin{aligned} Y_0 &= H_0 X_0 + z^{-2} H_1 X_1 \\ Y_1 &= H_0 X_1 + H_1 X_0 = H_{0+1} X_{0+1} - H_0 X_0 - H_1 X_1 \end{aligned} \quad (3.1)$$

Where  $H_{i+j} = H_i + H_j$  and  $X_{i+j} = X_i + X_j$

This 2-parallel fast FIR filter contains 5 subfilters, however, the 2 terms  $H_0X_0$  and  $H_1X_1$  are common and can be shared for the computation of  $Y_0$  and  $Y_1$ . This low complexity 2-parallel FIR filter structure is shown in figure 3.1, which computes a block of 2 outputs using 3 distinct subfilters of length  $N/2$  and 4 pre/post processing addition operations. It requires  $3N/2$  multiplications and  $3(N/2 - 1) + 4$  additions as opposed to  $2N$  multiplications and  $2(N-1)$  additions in the traditional parallel FIR filter derived directly from polyphase decomposition. The subfilters  $H_0, H_1$  and  $H_0+H_1$  for  $N=6$  contain the filter coefficients of  $H=\{h_0, h_1, h_2, h_3, h_4, h_5\}$  as  $H_0=\{h_0, h_2, h_4\}$ ,  $H_1=\{h_1, h_3, h_5\}$  and  $H_{0+1}=\{h_0+h_1, h_2+h_3, h_4+h_5\}$ . It should be noted that the addition of  $H_0$  and  $H_1$  does not cost anything in terms of the implementation because the filter coefficients are fixed and known prior to the implementation. This sum can be computed off-line. This low complexity 2-parallel FIR filter requires 12 multiplications and 13 additions, as opposed to 16 multiplications and 14 additions required for traditional parallel FIR filter.

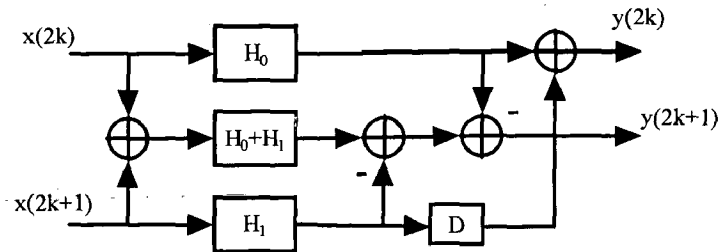


Fig 3.1 Reduced complexity 2-parallel fast FIR implementation

Since the implementational cost of a multiplier is much greater than that of an adder, the cost to implement the parallel filtering structure can be approximated as being proportional to the number of multipliers required for implementation. This is a very reasonable approximation for comparison purposes. Based upon this approximation, the 2-parallel fast FIR filtering structure requires about 25% less hardware (area) than the traditional 2-parallel implementation.

### 3.1.2 Matrix representation

The L- parallel FIR filter can be represented in matrix form as

$$Y_L = Q_L H_L P_L X_L \quad (3.2)$$

Where  $Y_L$  is an output matrix,  $Q_L$  is an post processing matrix which contain the post additions,  $P_L$  is an preprocessing matrix that determines the manner in which the inputs are combined,  $X_L$  is the input matrix and  $H_L$  is an  $L$  diagonal matrix. The entries of the diagonal matrix  $H_L$  are the subfilters of the parallel FIR filter. It should be noted that the application of FFA diagonalizes the pseudo-circulant matrix of (2.12).

The 2-parallel FIR filter can be represented in matrix form as

$$Y_2 = Q_2 H_2 P_2 X_2 \quad (3.3)$$

The 2-parallel fast FIR filter in (3.3) is represented in matrix form as

$$\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & z^{-2} \\ -1 & 1 & -1 \end{bmatrix} \text{diag} \begin{bmatrix} H_0 \\ H_0 + H_1 \\ H_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} \quad (3.4)$$

### 3.1.3 Parallel filters by transposition

Any parallel FIR filter can be used to derive another parallel equivalent structure by transpose operation. Generally the transposed architecture has the same hardware complexity, but different finite word length performance.

The transposition of  $L$ -parallel FIR filter can be obtained by transposing the equation (3.2)

$$Y_F = (Q_L H_L P_L)^T X_F = P_L^T H_L^T Q_L^T X_F \quad (3.5)$$

Where  $Y_F = [Y_{L-1} \ Y_{L-2} \ \dots \ Y_0]^T$ ,  $X_F = [X_{L-1} \ X_{L-2} \ \dots \ X_0]^T$  and  $P_L^T, Q_L^T$  are the post and preprocessing matrices which will determine the output and input combinations for the parallel FIR filters.  $H_L^T$  is an  $L$  diagonal matrix which contains the subfilters.

The transposition of 2-parallel fast FIR filter in (3.3) leads to another equivalent structure

$$Y_{2F} = P_2^T H_2^T Q_2^T X_{2F} \quad (3.6)$$

The 2-parallel fast FIR filter of (3.3) in transposition can be obtained as (3.6) and is shown in figure 3.2

$$\begin{bmatrix} Y_1 \\ Y_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \text{diag} \begin{bmatrix} H_0 \\ H_0 + H_1 \\ H_1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ z^{-2} & -1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_0 \end{bmatrix} \quad (3.7)$$

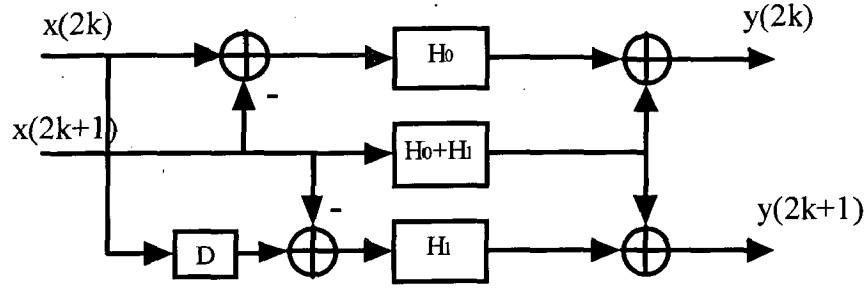


Fig 3.2 Transposed reduced complexity 2-parallel FIR filter

The transposed architecture can also be obtained by transposing the signal flow graph of the original parallel FIR filter. Generally both matrix transposition and signal flow graph transpositions are applicable to any FFA to generate equivalent parallel FIR filtering structures.

### 3.1.4 Three parallel fast FIR filter

A fast 3-parallel FIR algorithm can be derived by recursively applying a 2-parallel fast FIR algorithm.

$$\begin{aligned}
 Y &= Y_0 + z^{-1}Y_1 + z^{-2}Y_2 \\
 &= (X_0 + z^{-1}X_1 + z^{-2}X_2)(H_0 + z^{-1}H_1 + z^{-2}H_2) \\
 &= (X_0 + z^{-1}V)(H_0 + z^{-1}W)
 \end{aligned} \tag{3.8}$$

Where  $V = (X_1 + z^{-1}X_2)$  and  $W = (H_1 + z^{-1}H_2)$ . Using the fast FIR algorithm Y can be computed as

$$\begin{aligned}
 Y &= H_0X_0 + z^{-1}((H_0 + W)(X_0 + V) - H_0X_0 - VW) + z^{-2}VW \\
 &= [H_0X_0 + z^{-2}VW] + z^{-1}((H_0 + H_1 + z^{-1}H_2)(X_0 + X_1 + z^{-1}X_2) - H_0X_0 - VW)
 \end{aligned} \tag{3.9}$$

Where  $VW = (X_1 + z^{-1}X_2)(H_1 + z^{-1}H_2)$  can be computed as

$$\begin{aligned}
 VW &= (X_1 + z^{-1}X_2)(H_1 + z^{-1}H_2) \\
 &= [H_1X_1 + z^{-2}H_2X_2] + z^{-1}[(H_1 + H_2)(X_1 + X_2) - H_1X_1 - H_2X_2]
 \end{aligned} \tag{3.10}$$

Substituting equation(3.10) in equation(3.9) we get

$$\begin{aligned}
 Y &= [H_0X_0 + z^{-2}([H_1X_1 + z^{-2}H_2X_2] + z^{-1}[(H_1 + H_2)(X_1 + X_2) - H_1X_1 - H_2X_2])] \\
 &\quad + z^{-1}([(H_0 + H_1 + z^{-1}H_2)(X_0 + X_1 + z^{-1}X_2) - H_0X_0 - ([H_1X_1 + z^{-2}H_2X_2] + z^{-1}[(H_1 + H_2)(X_1 + X_2) - H_1X_1 - H_2X_2])])
 \end{aligned}$$

$$\begin{aligned}
&= H_0X_0 + z^{-1}[(H_0 + H_1)(X_0 + X_1) - H_0X_0 - H_1X_1] \\
&\quad + z^{-2}[H_1X_1 + (H_0 + H_1)X_2 + H_2(X_0 + X_1) - (H_1 + H_2)(X_1 + X_2) + H_1X_1 + H_2X_2] \\
&\quad + z^{-3}[(H_1 + H_2)(X_1 + X_2) - H_1X_1 - H_2X_2 + H_2X_2 - H_2X_2] + z^{-4}H_2X_2 \\
&= H_0X_0 - z^{-3}H_2X_2 + z^{-3}[(H_1 + H_2)(X_1 + X_2) - H_1X_1] \\
&\quad + z^{-1}([(H_0 + H_1)(X_0 + X_1) - H_1X_1] - (H_0X_0 - z^{-3}H_2X_2)) \\
&\quad + z^{-2}([(H_0 + H_1 + H_2)(X_0 + X_1 + X_2)] - [(H_0 + H_1)(X_0 + X_1) - H_1X_1] - [(H_1 + H_2)(X_1 + X_2) - H_1X_1])
\end{aligned} \tag{3.11}$$

The resulting 3-parallel FIR filter is given by

$$\begin{aligned}
Y_0 &= H_0X_0 - z^{-3}H_2X_2 + z^{-3}[(H_1 + H_2)(X_1 + X_2) - H_1X_1] \\
Y_1 &= z^{-1}([(H_0 + H_1)(X_0 + X_1) - H_1X_1] - (H_0X_0 - z^{-3}H_2X_2)) \\
Y_2 &= z^{-2}([(H_0 + H_1 + H_2)(X_0 + X_1 + X_2)] - [(H_0 + H_1)(X_0 + X_1) - H_1X_1] - [(H_1 + H_2)(X_1 + X_2) - H_1X_1])
\end{aligned} \tag{3.12}$$

The matrix form of 3-parallel FIR filter can be expressed as

$$Y_3 = Q_3 H_3 P_3 X_3$$

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} 1 & -z^{-3} & -z^{-3} & 0 & z^{-3} & 0 \\ -1 & -1 & z^{-3} & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 1 \end{bmatrix} \text{diag} \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_0 + H_1 \\ H_1 + H_2 \\ H_0 + H_1 + H_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} \tag{3.13}$$

The  $P_3$  and  $Q_3$  matrices are the pre-processing and post processing matrices respectively, while the  $H_3$  matrix is the diagonalized subfilter matrix. Hence, the 3-parallel FIR filter is constructed using 6 subfilters of length  $N/3$ , including  $H_0X_0, H_1X_1, H_2X_2, (H_0 + H_1)(X_0 + X_1), (H_1 + H_2)(X_1 + X_2), (H_0 + H_1 + H_2)(X_0 + X_1 + X_2)$ , and 3 pre processing and 7 post processing additions as shown in figure 3.3 . The overall computation requirement includes  $2N$  multiplications and  $2N+4$  additions. Comparing to the cost of the traditional and reduced complexity 3-

parallel structures, it is clear that the reduced complexity filtering structure provides a savings of approximately 33% traditional structure.

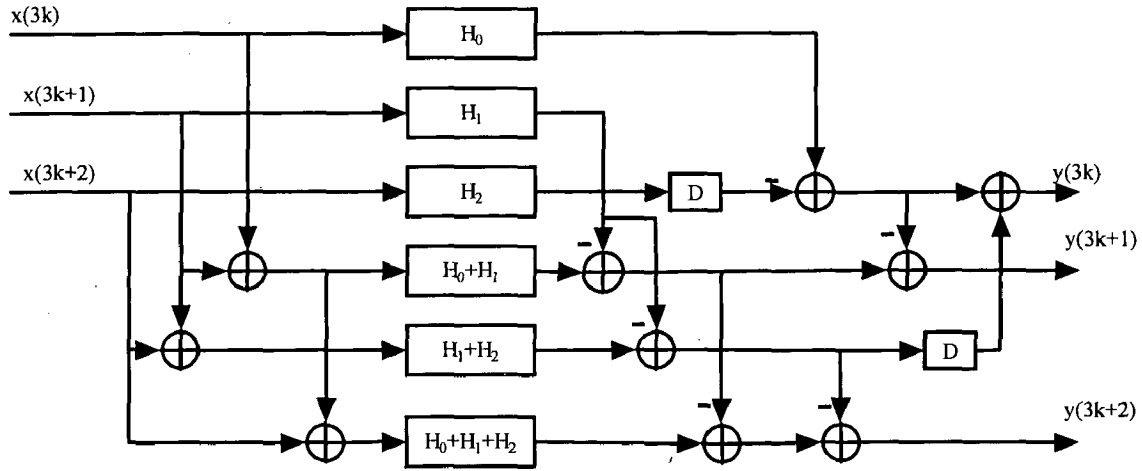


Fig 3.3 Reduced complexity 3-parallel FIR implementation

### 3.1.5 Fast Parallel FIR algorithm for large block sizes

Parallel FIR filters with long block sizes can be designed by cascading smaller length fast parallel filters i.e., an  $m$ -parallel FFA can be cascaded with an  $n$ -parallel FFA to produce an  $(m \times n)$  parallel FFA parallel filtering structure. The set of FIR filters that result from the application of the  $m$ -parallel FFA are further decomposed, one at a time by the application of  $n$ -parallel FFA. The resulting set of filters will be of length  $N/(m \times n)$ . When cascading the FFA's, it is important to keep track of both the number of multiplications and the number of additions required for the filtering structure. The number of required multiplications for an  $L$ -parallel FIR filter with  $L=L_1L_2...L_r$  is given by

$$M = \frac{N}{\prod_{i=1}^r L_i} \prod_{i=1}^r M_i \quad (3.14)$$

Where  $r$  is the number of FFA's used,  $L_i$  is the block size of FFA at step- $i$ ,  $M_i$  is the number of filters that result from the application of the  $i^{th}$  FFA and  $N$  is the length of the filter. The number of required adders can be calculated as follows

$$A = A_1 \prod_{i=2}^r L_i + \sum_{i=2}^r \left[ A_i \left( \prod_{j=i+1}^r L_j \right) \left( \prod_{k=1}^{i-1} M_k \right) \right] + \left( \prod_{i=1}^r M_i \right) \left( \frac{N}{\prod_{i=1}^r L_i} - 1 \right) \quad (3.15)$$

Where  $A_i$  is the number of pre/post processing adders required by the  $i^{\text{th}}$  FFA. Consider the case of cascading two 2-parallel FFA's to obtain a 4-parallel FIR structure as shown in figure 3.4. The resulting 4-parallel filtering structure would require  $9N/4$  multiplications and  $20+9(N/4 - 1)$  additions for implementation. The reduced complexity 4-parallel filtering structure represents an hardware (area) savings of nearly 44% when compared to the  $4N$  multiplications required in the traditional 4-parallel filtering structure.

In matrix form, the reduced complexity 4-parallel FIR filter is represented as follows

$$Y_{4P} = B_4 (I_{3 \times 3} \otimes Q_4) H_4 (P_2 \otimes P_2) X_{4P} \quad (3.16)$$

$$\text{Where } Y_{4P} = \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}, B_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & z^{-4} \\ 0 & 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & -1 \end{bmatrix}, Q_4 = \begin{bmatrix} 1 & 0 & z^{-4} \\ -1 & 1 & -1 \end{bmatrix}$$

$$H_4 = \text{diag} \begin{bmatrix} H_0 \\ H_0 + H_2 \\ H_2 \\ H_0 + H_1 \\ H_0 + H_1 + H_2 + H_3 \\ H_2 + H_3 \\ H_1 \\ H_1 + H_3 \\ H_3 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } X_{4P} = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

Note that  $X_{4P}$  and  $Y_{4P}$  are permuted versions of  $X_4$  and  $Y_4$  respectively.  $B_4$  can be obtained from  $Q_2$  by replacing 1 by  $I_{2 \times 2}$ , 0 by  $0_{2 \times 2}$  and by appropriate unfolding[4] of the delay operator  $z^{-2}$  of the 2-parallel FIR filter. The  $\otimes$  is the tensor or Kronecker product operator[7]. The tensor product is extremely useful in signal processing applications because it allows large matrices to be represented by small matrices. Using the tensor

product, the reduced complexity of 4-parallel FIR filter in relatively compact form can be represented. It should be noted that the  $Q_4$  matrix is essentially identical to the  $Q_2$  matrix of the 2-parallel FFA with the only difference being the power delay operator in the matrices. The 4-parallel FIR filter structure shown in fig 3.4 can be thought of as 3 separate 2-parallel FFAs each producing 2 outputs, which are combined by  $B_4$  to produce the 4 filter outputs.

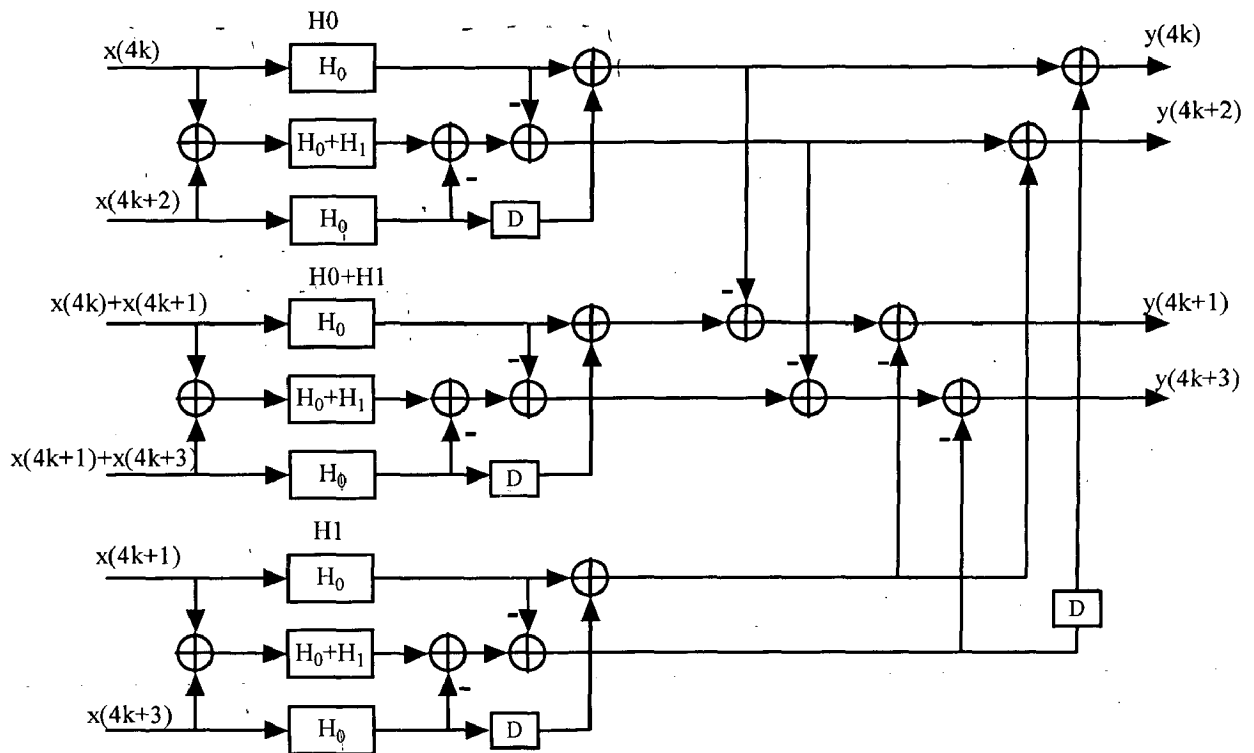


Fig 3.4 4-parallel fast FIR filter by cascading two 2-parallel FFA's( $F_0$  and  $F_1$ )

### 3.2 Fast FIR filters based on Frequency Spectrum

In [25], it was shown that the power consumption of arithmetic units can be reduced if statistical properties of the input signal is exploited. In [26], it is shown that the hardware cost can be reduced by exploiting the frequency spectrum characteristics of the given transfer function. This is achieved by selecting appropriate FFA structures out of many possible FFA structures all of whom have similar hardware complexity at the word-level. However, their complexity can differ significantly at the bit-level. For example, in narrowband low-pass filters, the signs of consecutive unit sample response values do not



change much and therefore their difference can require fewer number of bits than their sum. This favors the use of a parallel structure which requires subfilters which require difference of consecutive unit sample response values as opposed to sum. In addition to the appropriate selection of FFA structures, proper quantization of subfilters is important for low-power or low hardware cost implementation of parallel FIR filters.

### 3.2.1 FFA structures for 2-parallel and 3-parallel FIR filters

By an simple modification of 2-parallel FIR filter in (3.1), the following FFA1 is derived

$$\begin{aligned} Y_0 &= H_0 X_0 + z^{-2} H_1 X_1 \\ Y_1 &= -H_{0-1} X_{0-1} + H_0 X_0 + H_1 X_1 \end{aligned} \quad (3.17)$$

Where  $H_{i-j} = H_i - H_j$  and  $X_{i-j} = X_i - X_j$ .

The structure derived by FFA1 [6] is shown in figure 3.5 where as the FFA0 structure is in figure 3.4.

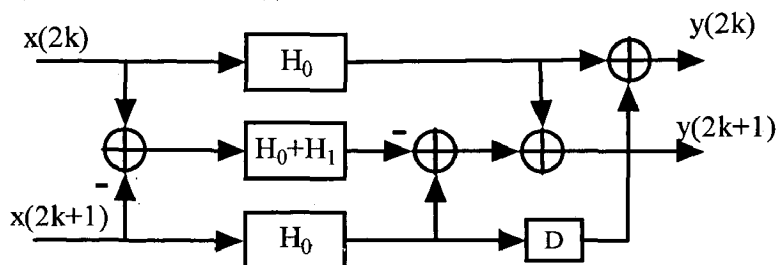


Fig 3.5 FFA1 structure of 2-parallel fast FIR filter

The structures derived by FFA0 and FFA1 are essentially the same except some sign changes. Notice that, in FFA1,  $H_{0-1}$  is used instead of  $H_{0+1}$ . When an FIR filter is implemented using a multiplierless approach, the hardware complexity is directly proportional to the number of nonzero bits in the filter coefficients. If the signs of the given impulse response sequences do not change frequently as in the narrowband low-pass filter cases, the coefficient magnitudes of  $H_0 + H_1$  are likely to be larger than those of  $H_0 - H_1$ . Then,  $H_0 + H_1$  has more nonzero bits in the coefficients than  $H_0 - H_1$  [5]. If the signs of the given impulse response sequences change frequently as in the wide-band low-pass filter cases,  $H_0 - H_1$  is likely to have more nonzero bits in the coefficients than  $H_0 + H_1$ . Thus, to achieve minimum hardware cost, it is necessary to select either FFA0 or FFA1 depending upon the frequency spectrum specifications.

The  $(3 \times 3)$  FFA produces a parallel filtering structure of block size 3. With  $L=3$ , we have

$$\begin{aligned}
Y_0 &= H_0 X_0 + z^{-3} (H_1 X_2 + H_2 X_1) \\
Y_1 &= H_0 X_1 + H_1 X_0 + z^{-3} (H_2 X_2) \\
Y_2 &= H_0 X_2 + H_1 X_1 + H_2 X_0
\end{aligned}
\tag{3.18}$$

This can be written in FFA0 form as shown in equation (3.12). This structure computes a block of 3 outputs using 6 length  $N/3$  FIR filters and 10 preprocessing and postprocessing additions, which requires  $6(N/3)$  multipliers and  $6(N/3 - 1) + 10$  adders. Notice that  $(3 \times 3)$  FFA0 structure provides a saving of approximately 33% over the traditional structure. The  $(3 \times 3)$  FFA1 structure can be obtained by modifying (3.18) as follows:

$$\begin{aligned}
Y_0 &= H_0 X_0 + z^{-3} (H_2 X_2) - z^{-3} (H_{2-1} X_{2-1} - H_1 X_1) \\
Y_1 &= -(H_{0-1} X_{0-1} - H_1 X_1) + (H_0 X_0 + z^{-3} H_2 X_2) \\
Y_2 &= H_{0-1+2} X_{0-1+2} - (H_{0-1} X_{0-1} - H_1 X_1) - (H_{2-1} X_{2-1} - H_1 X_1)
\end{aligned}
\tag{3.19}$$

Figure 3.6 shows the filtering structure that results from the  $(3 \times 3)$  FFA1. The following  $(3 \times 3)$  FFA2 structure given by (3.20), which is efficient when the coefficient magnitudes of  $H_{0-2}$  are smaller than those of  $H_{0-1+2}$  or  $H_{0+1+2}$ .

$$\begin{aligned}
Y_0 &= H_0 X_0 + z^{-3} (H_2 X_2 - H_{2-1} X_{2-1} + H_1 X_1) \\
Y_1 &= -H_{0-1} X_{0-1} + H_1 X_1 + H_0 X_0 + z^{-3} H_2 X_2 \\
Y_2 &= -H_{0-2} X_{0-2} + H_0 X_0 + H_1 X_1 + H_2 X_2
\end{aligned}
\tag{3.20}$$

Figure 3.7 shows the resulting FFA2 structure for 3-parallel FIR filter.

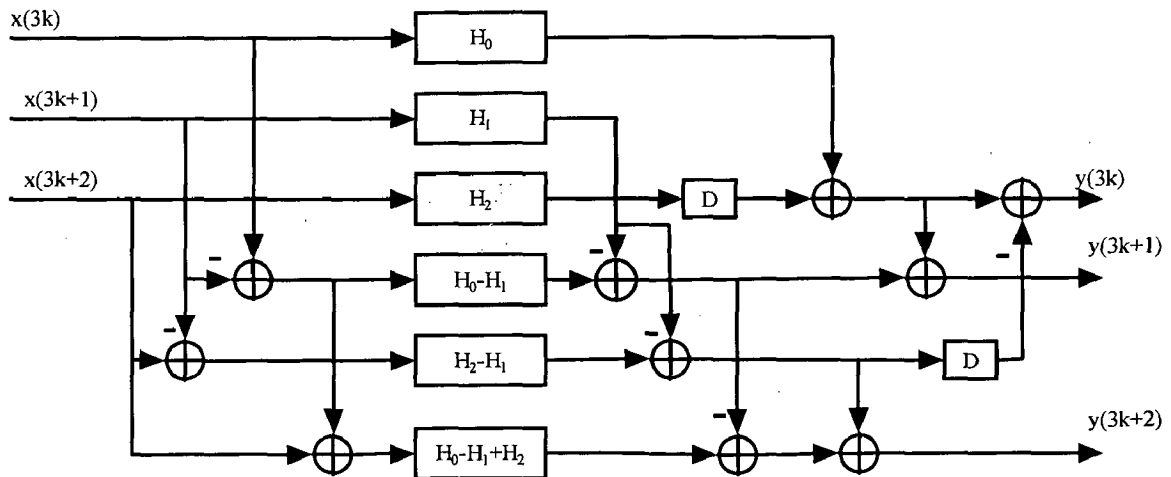


Fig 3.6 FFA1 structure for 3-parallel FIR filter

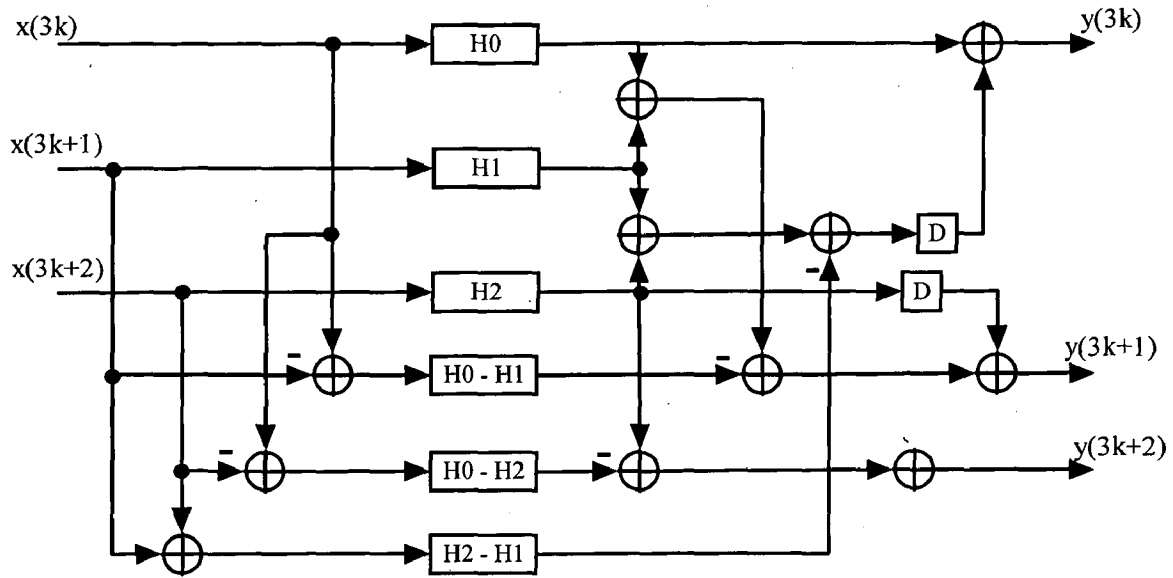


Fig 3.7 FFA2 structure for 3-parallel FIR filter

### 3.2.2 Cascading FFA's

The  $(2 \times 2)$  and  $(3 \times 3)$  FFAs can be cascaded together to achieve higher levels of parallelism. The cascading of FFAs is a straightforward extension of the original FFA application [2]. For example, the  $(4 \times 4)$  FFA can be obtained by first applying the  $(2 \times 2)$  FFA0 to (2.1) and then applying the  $(2 \times 2)$  FFA0 or the  $(2 \times 2)$  FFA1 to each of the filtering operations that result from the first application of the FFA0. The resulting  $(4 \times 4)$  FFA structure is shown in figure 3.8. Each filter block  $F_0$ ,  $F_0+F_1$ , and  $F_1$  represents a  $(2 \times 2)$  FFA structure and can be replaced separately by either  $(2 \times 2)$  FFA0 or  $(2 \times 2)$  FFA1. Each filter block  $F_0$ ,  $F_0 + F_1$ , and  $F_1$  is composed of three subfilters as follows:

- (i)  $F_0 : H_0, H_2, H_0 \pm H_2$ ,
- (ii)  $F_0 + F_1 : H_0 + H_1, H_2 + H_3, (H_0 + H_1) \pm (H_2 + H_3)$ ,
- (iii)  $F_1 : H_1, H_3, H_1 \pm H_3$ ,

where

$$\pm = \begin{cases} +, & \text{for FFA0,} \\ -, & \text{for FFA1.} \end{cases} \quad (3.21)$$

When the filter block  $F_0 + F_1$  is implemented using FFA1 structure, the subfilters are  $H_{0+1}$ ,  $H_{2+3}$ , and  $H_0 + H_1 - H_2 + H_3$ . Thus, even though FFA1 structure is used for slowly varying impulse response sequences, optimum performance is not guaranteed. In this

case, better performance can be obtained by using the FFA1' shown in figure 3.9. Since the subfilters in FFA1' are  $H_{0-1}$ ,  $H_{2-3}$ , and  $H_{0-1} - H_{2-3}$ , the FFA1' gives smaller number of nonzero bits than FFA1 for the case of slowly varying impulse response sequences. Notice that the FFA1' structure can be derived by first applying the  $(2 \times 2)$  FFA1 (instead of the  $(2 \times 2)$  FFA0) to (2.1). When the filter block  $F_0 + F_1$  in Figure 3.8 is replaced by FFA1' in Figure 3.9, it can be shown that the outputs are  $y(4k)$ ,  $-y(4k + 1)$ ,  $y(4k + 2)$ , and  $-y(4k + 3)$ . The complexity computation is similar to the fast parallel FIR filter.

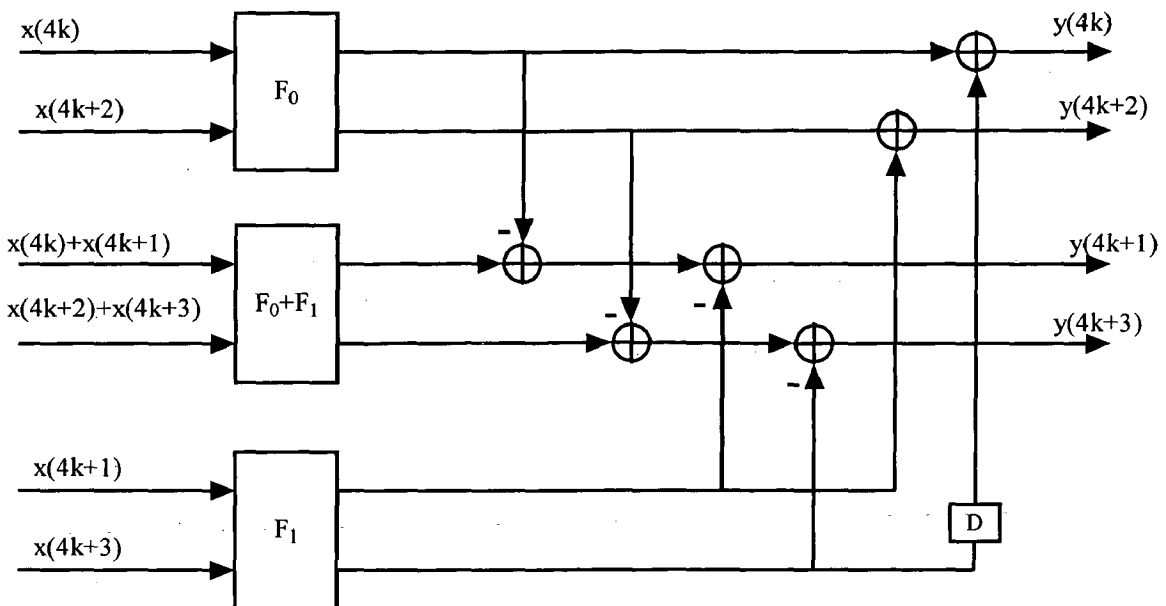


Fig 3.8 FFA0 or FFA1 4-parallel FIR filter structure

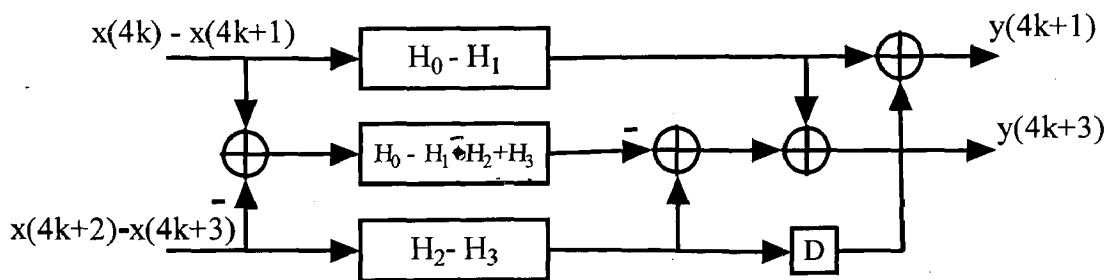


Fig 3.9 FFA1' structure of 4-parallel FIR filter

### 3.2.3 Selection of FFA types

For given length  $N$  unit sample response values  $\{h_i\}$  and block size  $L$ , the selection of best FFA type can be roughly determined by comparing the signs of the values in subfilters  $H_0, H_1, \dots, H_{L-1}$ . For example, in the case of  $L = 2$  and even  $N$ ,  $H_0$ , and  $H_1$  are

$$\begin{aligned} H_0 &= h_0, h_2, \dots, h_{N-2}, \\ H_1 &= h_1, h_3, \dots, h_{N-1}. \end{aligned} \quad (3.22)$$

From (3.22), the  $i^{\text{th}}$  value of  $H_0$  can be paired with the  $i^{\text{th}}$  value of  $H_1$  as  $(h_0, h_1), (h_2, h_3), \dots, (h_{N-2}, h_{N-1})$ . Comparing the signs of the values in each pair, the number of pairs with opposite signs and the number of pairs with the same signs can be determined. If the number of pairs with opposite signs is larger than the number of pairs with the same signs,  $H_0 + H_1$  is likely to be more efficient than  $H_0 - H_1$ . The sign-comparing procedure can be extended to any block size of  $L$  with appropriate modifications.

## 4 PARALLEL FIR FILTER STRUCTURES BASED ON FAST CONVOLUTION ALGORITHMS

In this chapter, an approach to further improve the throughput of FIR filters can be done with short length linear convolution algorithms. A set of fast short length linear convolution algorithms has been developed to realize parallel processing of FIR filters [8].

### 4.1 Fast Parallel FIR filters based on linear convolution

Any  $L \times L$  convolution algorithm can also be used to derive an  $L$ -parallel fast filter structure. Generally, the  $L$ -parallel FIR filter

$$Y = HX \quad (4.1)$$

is first expressed as

$$[Y_0 \ Y_1 \ \dots \ Y_{L-1}]^T = \begin{bmatrix} H_{L-1} & H_{L-2} & \dots & H_1 & H_0 & 0 & \dots & 0 \\ 0 & H_{L-1} & \dots & H_2 & H_1 & H_0 & \dots & 0 \\ 0 & 0 & \dots & H_3 & H_2 & H_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & H_{L-1} & H_{L-2} & \dots & H_0 \end{bmatrix} \begin{bmatrix} z^{-L} X_1 \\ z^{-L} X_2 \\ \dots \\ z^{-L} X_{L-1} \\ X_0 \\ X_1 \\ \dots \\ X_{L-1} \end{bmatrix} \quad (4.2)$$

This form of the standard parallel filtering algorithm is similar to the transpose form of a linear convolution [1]. Using this idea, reduced complexity of parallel filtering can be generated. The basic idea is to start with an optimal linear convolution and take its transposition to generate the parallel filtering algorithm. Some efficient fast short convolutions based on Winograd algorithm, which have less number of multiplications than FFAs, are shown in Appendix C. However, when the convolution length increases, the number of additions increases dramatically, which leads to complex pre addition and post addition matrices that are not practical for hardware implementation. Therefore, if we could use fast convolution algorithms to decompose the convolution matrix with simple preaddition and post addition matrices, we can get computationally efficient parallel FIR filter with reduced number of delay elements. Fortunately, the mixed radix

algorithm in [7], which decomposes the convolution matrix with tensor product into two short convolutions can be used. This algorithm is combined with fast two and three point convolution algorithms to obtain a general Iterated Short Convolution Algorithm (ISCA). Although fast convolution of any length can be derived from Cook–Toom algorithm or Winograd algorithm [4], their pre addition or post addition matrices may contain elements not in the set  $\{1,0,-1\}$  (i.e., shown in Appendix C), which sometimes make them not suitable for hardware implementation of iterated convolution algorithm.

## 4.2 ITERATED SHORT CONVOLUTION ALGORITHM (ISCA) [7]

A long convolution can be decomposed into several levels of short convolutions. After fast convolution algorithms for short convolutions are constructed, they can be iteratively used to implement the long convolution [4]. The mixed radix algorithm [7] is used to derive the generalized iterated short convolution algorithm using the Tensor Product operator in matrix form.

A  $M \times M$  convolution can be decomposed into  $m \times m$  convolution and a  $n \times n$  convolution, whose short convolution algorithms can be constructed with fast convolution algorithms such as Cook–Toom algorithm[4] or Winograd algorithm [4][11] and represented as  $S_{2m-1} = Q_m H_m P_m X_m$  and  $S_{2n-1} = Q_n H_n P_n X_n$  respectively.  $Q_m$  and  $Q_n$  are post addition matrices.  $P_m$  and  $P_n$  are pre addition matrices.  $H_m$  and  $H_n$  are diagonal matrices, which can be denoted as  $H_m = \text{diag} \left[ P_m \times [h_0 \ h_1 \ \dots \ h_{m-1}]^T \right]$  and  $H_n = \text{diag} \left[ P_n \times [h_0 \ h_1 \ \dots \ h_{n-1}]^T \right]$  respectively. They determine the number of required multiplications in the iterated short convolution algorithm.  $X_m$  and  $[h_0 \ h_1 \ \dots \ h_{m-1}]^T$  are two column vectors, containing the two input sequences for  $m \times m$  convolution..  $X_n$  and  $[h_0 \ h_1 \ \dots \ h_{n-1}]^T$  are two column vectors, containing the two input sequences for  $n \times n$  convolution. These two convolutions result in two outputs:  $S_{2m-1}$  and  $S_{2n-1}$  of length  $(2m-1)$  and  $(2n-1)$  respectively.

Using the mixed radix algorithm [7], the resulting iterated short convolution algorithm can be represented as

$$S_{2M-1} = A_{M\_mn} (Q_m \otimes Q_n) H_{M\_mn} (P_m \otimes P_n) X_M \quad (4.3)$$

Where  $H_{M\_mn} = \text{diag} \left[ (P_m \otimes P_n) \times [h_0 \ h_1 \ \dots \ h_{M-1}]^T \right]$  and  $A_{M\_mn}$  is a  $(2M-1)$  by  $((2m-1)(2n-1))$  matrix, composed of  $(2n-1)$  by  $(2n-1)$  unity matrices as shown in equation (4.4)

$$A_{M\_mn} = \left[ \begin{array}{c} \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{matrix}} \\ \underbrace{\hspace{10em}}_{I_{(2n-1) \times (2n-1)}} \end{array} \right] \dots \left[ \begin{array}{c} \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{matrix}} \\ \underbrace{\hspace{10em}}_{I_{(2n-1) \times (2n-1)}} \end{array} \right] \dots \left[ \begin{array}{c} \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{matrix}} \\ \underbrace{\hspace{10em}}_{I_{(2n-1) \times (2n-1)}} \end{array} \right] \dots \left[ \begin{array}{c} \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{matrix}} \\ \underbrace{\hspace{10em}}_{I_{(2n-1) \times (2n-1)}} \end{array} \right] \quad (4.4)$$

The order of convolution can be of any order, first i.e.,  $m \times m$  first followed by  $n \times n$  convolution or  $n \times n$  convolution followed by  $m \times m$  convolution to obtain  $M \times M$  ( $mn$  or  $nm$ ) convolution. In this way, any long convolution can be decomposed can be decomposed into any combination of fast short convolutions. A  $L \times L$  ( $L=L_1.L_2...L_r$ ) linear convolution can be decomposed into  $r$  short convolutions  $S_{2L_i-1} = Q_{L_i} H_{L_i} P_{L_i} X_{L_i}$  ( $i = 1, 2, \dots, r$ ). One of the resulting iterated convolution algorithm can be represented by

$$S_{2L-1} = A_L \left( Q_{L_1} \otimes \left( \dots \left( Q_{L_{r-1}} \otimes Q_{L_r} \right) \right) \right) H_L \left( P_{L_1} \otimes \left( \dots \left( P_{L_{r-1}} \otimes P_{L_r} \right) \right) \right) X_L \quad (4.5)$$

Where  $H_L = \text{diag} \left( \left( P_{L_1} \otimes \left( \dots \left( P_{L_{r-1}} \otimes P_{L_r} \right) \right) \right) \times [h_0 \ h_1 \ \dots \ h_{L-1}]^T \right)$  and  $A_L$  can be computed using the following procedure



$$\begin{aligned}
A_1 &= A_{L_1 L_2} \\
A_2 &= A_{L_1 L_2 L_3} \cdot (A_1 \otimes I_{(2L_3-1) \times (2L_3-1)}) \\
&\dots\dots \\
A_{r-1} &= A_{L_1 L_2 L_3 \dots L_r} \cdot (A_1 \otimes I_{(2L_r-1) \times (2L_r-1)}) \\
A_L &= A_{r-1}
\end{aligned} \tag{4.6}$$

The above equation (4.4) is the iterated short convolution algorithm. The mixed radix algorithm [7] combines two short convolutions to get a longer convolution, while this iterated short convolution algorithm can combine any numbers of short convolutions and thus it is more efficient.

The iterated convolution structure can be transposed to obtain a fast parallel FIR filter. An  $L(L=L_1 L_2 \dots L_r)$  parallel  $N$ -tap FIR filter based on iterated  $L_i \times L_i$  convolutions,  $S_{2L_i-1} = Q_{L_i} H_{L_i} P_{L_i} X_{L_i}$  ( $i = 1, 2, \dots, r$ ), can be expressed as

$$Y_L = P^T H_L Q^T A^T X_L \tag{4.7}$$

Where  $Y_L = [Y_{L-1} \ Y_{L-2} \ \dots \ Y_0]^T$ ;

$$X_L = [X_{L-1} \ \dots \ X_1 \ X_0 \ z^{-L} X_{L-1} \ \dots \ z^{-L} X_2 \ z^{-L} X_1]^T;$$

$X_i$  ( $i = 0, 1, \dots, L-1$ ) inputs  $x_{Lk+i}$ , ( $k = 0, 1, 2, \dots, r$ )

$$H_L = \text{diag} \left[ P \times [H_0 \ H_1 \ \dots \ H_{L-1}]^T \right]$$

$$P = \left( P_{m_1 \times n_1} \otimes \left( \dots \left( P_{m_{r-1} \times n_{r-1}} \otimes P_{m_r \times n_r} \right) \right) \right)$$

$H_i$  ( $i = 0, 1, \dots, L-1$ ) subfilters containing the coefficients  $h_{Lk+i}$ , ( $k = 0, 1, 2, \dots, r$ )

$$P^T = \left( P_{m_1 \times n_1}^T \otimes \left( \dots \left( P_{m_{r-1} \times n_{r-1}}^T \otimes P_{m_r \times n_r}^T \right) \right) \right)$$

$$Q^T = \left( Q_{m_1 \times n_1}^T \otimes \left( \dots \left( Q_{m_{r-1} \times n_{r-1}}^T \otimes Q_{m_r \times n_r}^T \right) \right) \right)$$

$A^T$  is the transpose of the matrix defined by (4.4)

$P^T$  and  $Q^T$  are the preprocessing and post processing matrices, which determine the manner in which the inputs and outputs are combined, respectively [4].

Let us illustrate this through an example of 4-parallel FIR filter structure, implemented by  $4 \times 4$  convolution, which is done by the iterated two  $2 \times 2$  convolutions as

$$Y_4 = (P_2^T \otimes P_2^T) H_4 (Q_2^T \otimes Q_2^T) A_{4\_22}^T X_4$$

$$\text{Where } A_{4\_22}^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} Q_2^T \otimes Q_2^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$P_2^T \otimes P_2^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix}$$

$$H_4 = \text{diag}([H_0 \quad H_0 - H_1 \quad H_1 \quad H_0 - H_1 \quad H_0 - H_1 - H_2 + H_3 \quad H_1 - H_3 \quad H_2 \quad H_2 - H_3 \quad H_3])$$

$$X_4 = [X_3 \quad X_2 \quad X_1 \quad X_0 \quad z^{-4}X_3 \quad z^{-4}X_2 \quad z^{-4}X_1]^T$$

This 4-parallel FIR filter architecture is as shown in figure 4.1. Similarly a 6-parallel FIR filter can be implemented by  $2 \times 2$  and  $3 \times 3$  convolution. The order must be important as it reduces the number of adders in its implementation. For ISCA, higher order convolution should be first followed by lower order convolution for less number of adders to be implemented.

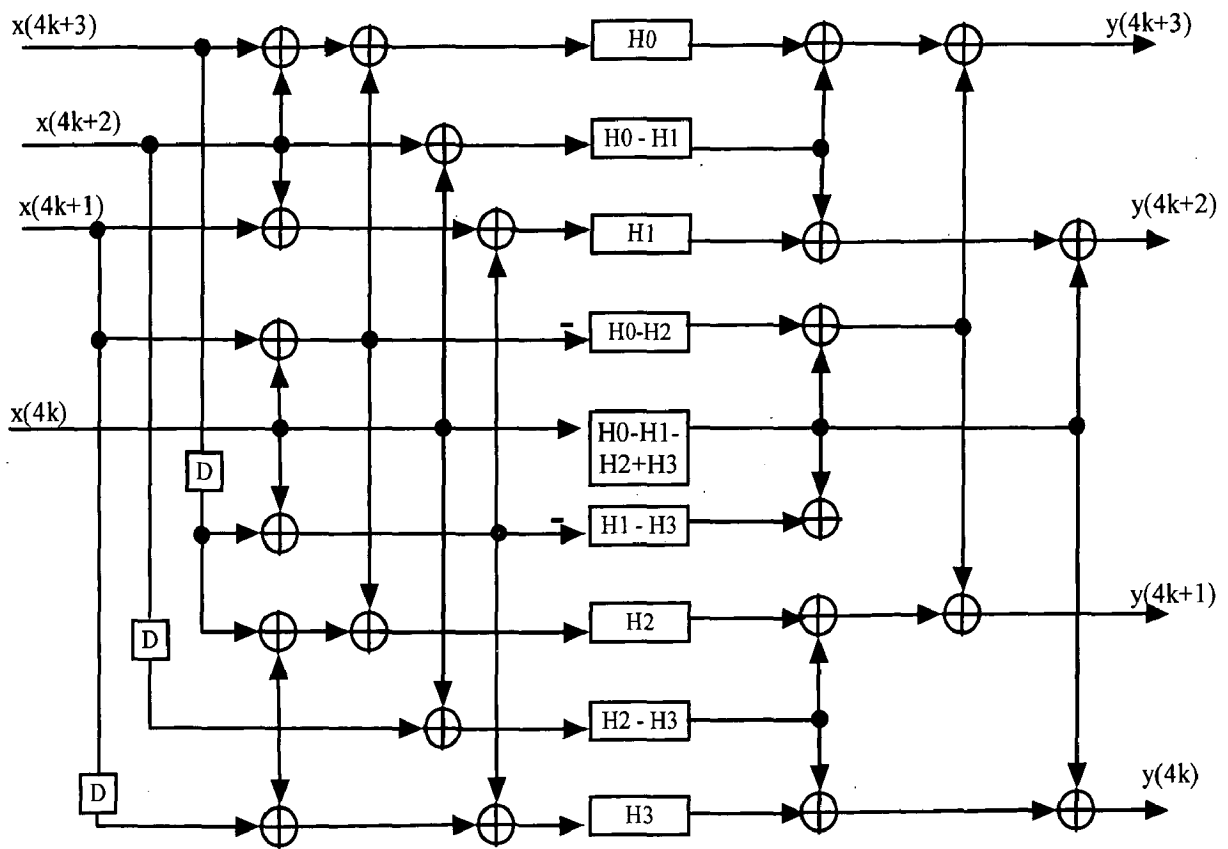


Fig 4.1 ISCA based 4-parallel FIR filter

#### 4.2.1 Complexity Computation:

The number of required multiplications is determined by the diagonal matrix  $H_L$  in (4.7) is given by

$$M = \frac{N}{\prod_{i=1}^r L_i} \prod_{i=1}^r M_i \quad (4.8)$$

Where  $r$  is the number of  $L_i \times L_i$  convolutions used,  $M_i$  is the number of multiplications used in the  $L_i \times L_i$  convolution, which is determined by  $H_L$ , and  $N$  is the length of the original filter. All multiplications lie in subfilters of the same length. The number of subfilters is determined by  $\prod_{i=1}^r M_i$ , and the length of each subfilter is given by  $N / \prod_{i=1}^r L_i$ .

The number of required adders is determined by the adders used in pre and post processing matrices and the adders used in subfilters, is given by

$$A = \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(P_{m_i \times n_i}^T) \right] + \prod_{i=1}^r M_i \left( \frac{N}{\prod_{i=1}^r L_i} - 1 \right) + \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(Q_{m_i \times n_i}^T) \right] \quad (4.9)$$

The first sum gives the preprocessing adders and the second sum gives the numbers of adders in the subfilters followed by the third one, which gives the post processing adders of the parallel FIR filter. Function  $A(P_{m_i \times n_i}^T)$  is the minimum number of adders in the matrix  $P_{m_i \times n_i}^T$ . Since each row of  $A_L^T$  has only one "1", it will not increase the number of adders. The number of required additions depend on the order of iteration.  $m \times m$  convolution is iterated ahead of convolution ( $m > n$ ), will lead to lowest adder complexity.

The number of required delay elements is counted by the  $(L-1)$  delay elements in the input side and the ones used in the subfilters, and is given by

$$D = L - 1 + \prod_{i=1}^r M_i \left( \frac{N}{\prod_{i=1}^r L_i} - 1 \right) \quad (4.10)$$

### 4.3 Improved Fast Parallel FIR filter structure

When  $L$  is large, ISCA-based parallel filter involves many subfilters, which require a large number of multiplication operations but the same hardware structure. Designing an efficient core to share the computation of all these subfilters in different time slots can reduce the hardware cost. Further reduction in the hardware i.e., no. of multiplications of fast parallel FIR filters can be done by transforming an FIR filter into linear convolution. Then an iterated short convolution is used to implement this linear convolution. This method will increase the no. of delay elements by decreasing the number of multipliers.

Simple control signals are used for the control of the data flow in the delay element matrix (DEM).

An  $N$ -tap FIR filter can be transformed into  $N \times N$  linear convolution as shown in figure 4.2. The last  $N-1$  rows of the  $2N-1$  outputs of  $N \times N$  linear convolution are summed with the upper  $N-1$  rows of the  $2N-1$  outputs of the following convolution to get the  $N$  outputs of the  $N$ -tap filter. Then the hardware cost will depend on the complexity of the  $N \times N$  linear convolution. The 'D' shown is an delay of  $N$  cycles for an  $N \times N$  linear convolution.

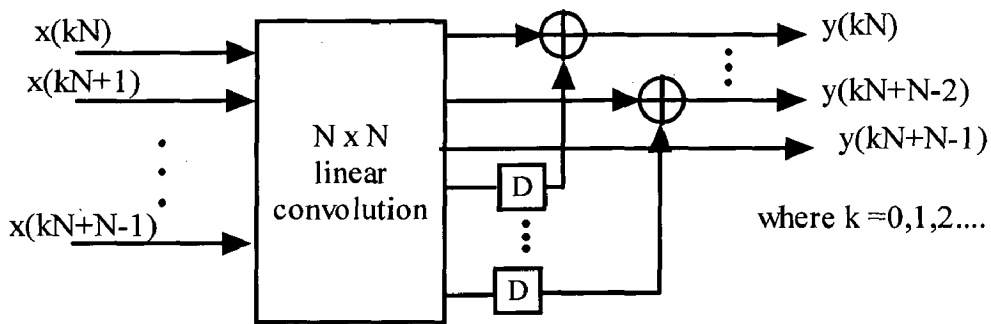


Fig 4.2 Implementation of  $N$ -tap FIR filter with  $N \times N$  linear convolution

Based on the above property the algorithm for hardware reduced parallel FIR filter is as follows:

#### 4.3.1 Improved structure Algorithm: [9]

Improved structures for a given  $L$ -parallel  $N$ -tap FIR filter can be obtained as:

1. Form an ISC based FIR filter by (4.7);
2. Replace the subfilters with a core  $(N/L) \times (N/L)$  linear convolution and two delay element matrices to arrange the input and output of the  $(N/L) \times (N/L)$  linear convolution;
3. Implement the  $(N/L) \times (N/L)$  linear convolution using the iterated short convolution algorithm.

Consider an example of 2-parallel 6-tap FIR filter to illustrate this algorithm.

$$Y_2 = P_2^T H_2 Q_2^T X_L \quad (4.11)$$

Where outputs  $Y_2 = [y(2k) \quad y(2k+1)]^T$

inputs  $X_2 = [x(2k+1) \quad x(2k) \quad x(2k-1)]^T ; k=0,1,2,\dots$

$$H_2 = \text{diag}[H_0 \quad H_0 - H_1 \quad H_1]$$

$$P_2 = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

The 2-parallel 6-tap FIR filter is as shown in figure 4.3.  $H_0$ ,  $H_0 - H_1$ ,  $H_1$  are three 3-tap subfilters of  $\{h_0, h_2, h_4\}$ ,  $\{h_0 - h_1, h_2 - h_3, h_4 - h_5\}$ ,  $\{h_1, h_3, h_5\}$  respectively.

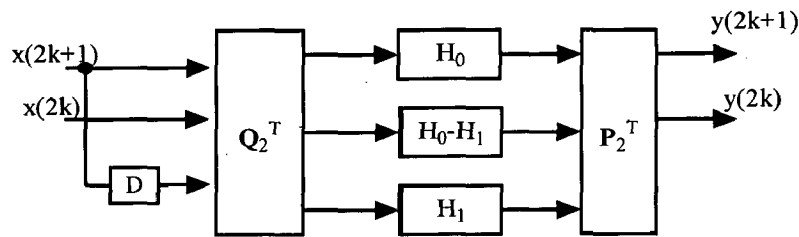


Fig 4.3 2-parallel 6-tap FIR filter

By transforming the computation of 3-tap FIR filter into that of  $3 \times 3$  linear convolution, an improved structure is obtained as shown in figure 4.4.

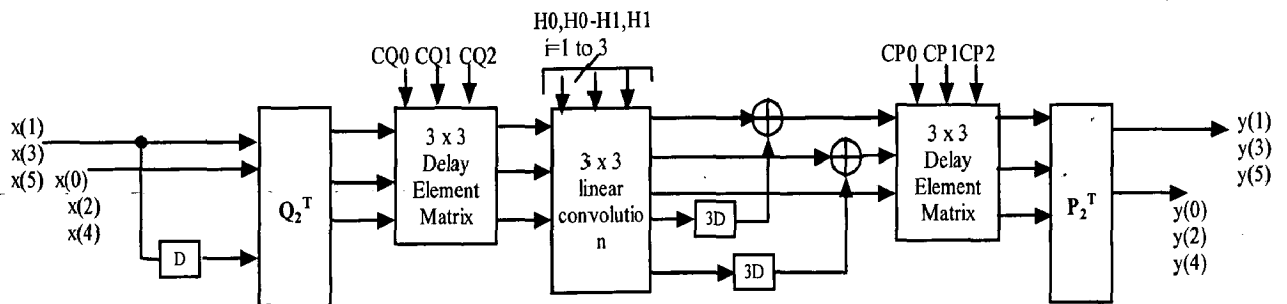


Fig 4.4 An improved 3-parallel structure for 2-parallel 6-tap FIR filter

The data flow of the Delay Element Matrix (DEM) in Fig. 4.5 is 'horizontal in, vertical out' or 'vertical in, horizontal out' and controlled by  $C_0$ ,  $C_1$  and  $C_2$  signals.  $C_0$  signal controls whether the data are 'horizontal' in or 'vertical in'.  $C_1$  signal controls whether the data are 'horizontal out' or 'vertical out'.  $C_2$  signal controls whether the data flow horizontally or vertically in the delay element matrix. The delay element function is as shown in figure 4.5. The structure can compute 6 input data in 3 clock cycles, thus it has the same throughput rate as previous 2-parallel FIR filter structure.

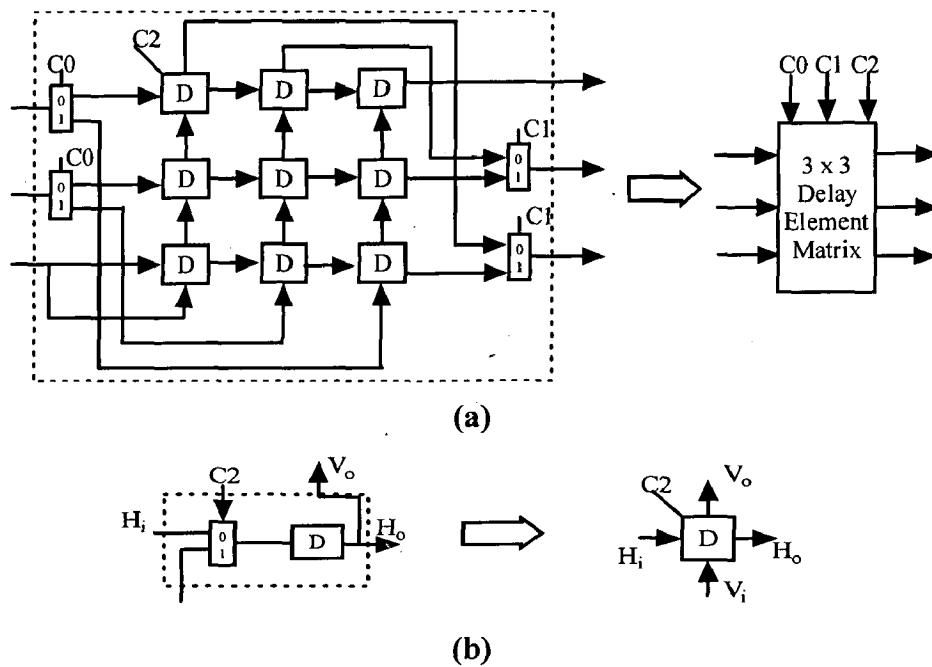


Fig 4.5 (a) 3x3 Delay Element Matrix (DEM) (b) Delay element function

### 4.3.2 Complexity Computation

For a  $L$ -parallel  $N$ -tap FIR filter, the computation is based on  $(N/L) \times (N/L)$  linear convolution, which can be implemented with the iterated short convolution algorithm in [5]. The number of required multiplications for the parallel FIR filter is equal to that of  $(N/L) \times (N/L)$  linear convolution. If  $N/L$  can be decomposed as  $L_1 L_2 L_3 \dots L_s$ , then the number of required multiplications can be given as:

$$M = \prod_{i=1}^s M_i \quad (4.12)$$

Where  $s$  is the number of  $L_i \times L_i$  convolutions used to implement  $(N/L) \times (N/L)$  linear convolution.  $M_i$  is the number of multiplications used in the  $L_i \times L_i$  convolution, which is determined by  $H_{L_i}$ . The number of required additions is made up of three parts: 1). additions used for the preprocessing and post processing matrices  $P^T$  and  $Q^T$  in (4.7); 2). additions used to implement the  $(N/L) \times (N/L)$  linear convolution by iterated short convolution; 3). additions used in Fig. 4.2 for transforming  $(N/L)$ -tap FIR filter into  $(N/L) \times (N/L)$  linear convolution, which can be given by  $(N/L)-1$ . Therefore, the total number of required addition can be given by:

$$\begin{aligned}
A = & \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(P_{m_i \times n_i}^T) \right] + \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(Q_{m_i \times n_i}^T) \right] \\
& + \sum_{i=1}^s \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^s m_k \right) A(P_{m_i \times n_i}) \right] + \sum_{i=1}^s \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^s m_k \right) A(Q_{m_i \times n_i}) \right]
\end{aligned} \quad (4.13)$$

Where  $r$  is the number of  $L_i \times L_i$  convolutions used to implement the  $L$ -parallel FIR filter. The number of required delay elements is also made up of three parts: 1)  $(L-1)$  delay elements in the input side as shown in Figure 4.3; 2). delay elements used in Figure 4.4 for transforming  $(N/L)$ -tap FIR filter into  $(N/L) \times (N/L)$  linear convolution, which can be given by  $\prod_{i=1}^r M_i \left( \frac{N}{L} - 1 \right)$ ; 3). delay elements used in the two delay element matrices, which can be noted as  $D_e$ , where  $D_e$  is given by

$$D_e = \prod_{i=1}^r M_i \cdot \frac{N}{L} + \min \left( \left( \prod_{i=1}^r M_i \right), \frac{N}{L} \right) \cdot \left| \left( \prod_{i=1}^r M_i \right) - \frac{N}{L} \right| \quad (4.14)$$

Where  $\prod_{i=1}^r M_i$  is actually the number of output of matrix  $Q^T$  in (4.7) and is also the number of input data that go into the delay element matrix before the  $(N/L) \times (N/L)$  linear convolution; it may be greater or less than  $N/L$ , and may be equal to  $N/L$  as in Fig. 4.4; when it is less than  $N/L$ , the shape of the delay element matrix is like Fig. 4.6(a); when it is greater than  $N/L$ , the shape of the Delay Element Matrix is like Fig. 4.6 (b).

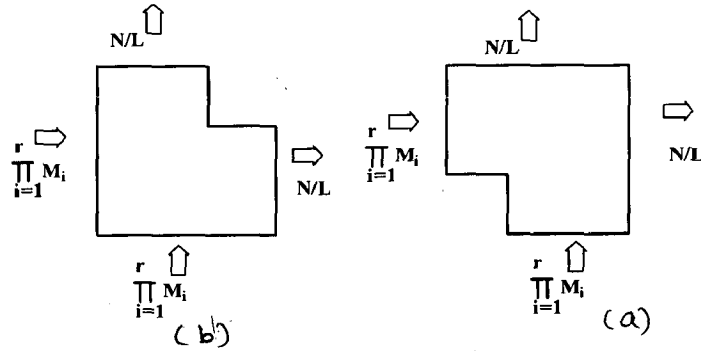


Fig 4.6 Shape of Delay Element Matrix (a)  $\prod_{i=1}^r M_i > N/L$  (b)  $\prod_{i=1}^r M_i < N/L$

Therefore, the total number of required delay elements can be given by:

$$D = \prod_{i=1}^r M_i \left( \frac{N}{L} - 1 \right) + 2D_e + L - 1 \quad (4.15)$$



## 5 Parallel FIR filters based on 2-stage parallelism

Although the idea of improved fast parallel structure is applied for parallel FIR filters, the computation of the subfilters is shared by a linear convolution processing core, its structure is irregular for some cases. If a  $L$ -parallel  $N$ -tap FIR filter contains subfilters of length  $(N/L)$  is assumed, then previous ISCA-based structures can process  $nL$  input samples in  $n$  clock cycles with all the  $n$  subfilters working simultaneously.  $n$  is also the number of output samples from the preprocessing matrix  $Q^T A^T$  in (4.7) when  $L$  input samples are input in each clock cycle, and thus the  $n$  subfilters will process  $n^2$  intermediate data, corresponding to the outputs of the preprocessing matrix, in  $n$  clock cycles. If the computation of these  $n^2$  intermediate data in  $n$  clock cycles is obtained by using one  $(N/L)$ -tap FIR filtering core, this core must be able to process  $n$  data in one clock cycle, i.e., the core must implement an  $n$ -parallel  $(N/L)$ -tap FIR subfilter. The hardware complexity of this  $n$ -parallel FIR subfilter is far less than that of the  $n$  subfilters, especially when  $N$  and  $L$  are large. This is the basic idea of the 2-stage parallel FIR filter structures. The improved structure for fast parallel FIR filter is an special case for 2-stage parallel FIR filter.

### 5.1 2-stage parallelism [8]

There are two methods for the parallel FIR filter based on 2-stage parallelism; 1) by replacing the  $(N/L) \times (N/L)$  linear convolution with  $(N/L)$  parallel FIR filter.

#### 5.1.1 Generalization (Method-1) of 2-stage Parallelism

The structures for a given  $L$ -parallel  $N$ -tap FIR filter can be generalized as follows.

- 1) Form an ISCA-based  $L$ -parallel FIR filter by (4.7).
- 2) Replace its subfilters with a second stage  $n$ -parallel FIR subfilter, where  $n$  is the number of subfilters involved in the first stage  $L$ -parallel implementation, and two DEMs of size  $n \times n$  each to arrange the input and output of the  $n$ -parallel FIR subfilter.
- 3) Implement the  $n$ -parallel FIR subfilter by first forming an ISCA-based  $n$ -parallel FIR filter from (4.7), and then replacing each delay element 'D' with 'nD'.

### 5.1.2 Method-2 of 2-stage parallelism

2). Direct application of the parallel FIR filter structures will have problems when  $L$  is large. Then the number of subfilters of the first stage  $L$ -parallel implementation,  $n$ , will increase dramatically. In this case, the number of required additions for preprocessing and post-processing matrices of the second stage  $n$ -parallel FIR subfilter will dominate the total number of required additions and lead to large amount of required additions. Furthermore, large  $n$  will also lead to a dramatic increase in the number of required delay elements because of the replacement of 'D' with ' $nD$ ' in the implementation of the second stage  $n$ -parallel FIR subfilter and the two DEMs of size  $n^2$ . Finally the latency of the design will also be long since the computation latency is  $2n$  clock cycles.

The improved implementation of  $L$ -parallel FIR filter structures (Method-2) by 2-stage parallelism can be generalized as

- 1) Form an ISCA-based  $L'$ -parallel FIR filter by (4.7), where  $L'$  is the first stage parallelism and it divides  $L$ .
- 2) Replace its subfilters with a second stage  $L''$ -parallel FIR subfilter, where  $L'' = (L/L')n$  and  $n$  is the number of subfilters involved in the first stage  $L'$ -parallel implementation, and  $2(L/L')$  DEMs of size  $n \times n$  needed to arrange the input and output of the  $n$ -parallel FIR subfilter.
- 3) Implement the  $L''$ -parallel FIR subfilter by first forming an ISCA-based  $L''$ -parallel FIR filter from (4.7), and then replacing each delay element " $D$ " with " $nD$ ".

### 5.1.3 Complexity Computation

For the  $L$ -parallel  $N$ -tap FIR filter, where  $L$  has only 2 and/or 3 as its prime factors, the number of subfilters of its first stage  $L'$ -parallel structure ( $L = L_1 \cdot L_2 \dots L_r$ ) can be given as

$$n = \prod_{i=1}^r M_i \quad (5.1)$$

Where  $r$  is the number of  $L_i \times L_i$  convolutions used,  $M_i$  is the number of multiplications used in the  $L_i \times L_i$  convolution, which is determined by  $H_{L_i}$  in (4.7). It is obvious that  $L'' = (L/L')n$  has only 2 and/or 3 as its prime factors and can be further decomposed as ( $L'' = L_1.L_2...L_s$ ). The number of subfilters of the second stage  $L''$ -parallel FIR subfilter can be given as:  $\prod_{i=1}^s M_i$ , which is also the total number of subfilters of the  $L$ -parallel  $N$ -tap FIR filter. The final subfilter length is  $\left(\frac{N}{nL'}\right)$ . Therefore the total number of required multiplications is given by

$$M = \left(\frac{N}{nL'}\right) \prod_{i=1}^s M_i \quad (5.2)$$

Where  $n$  and  $M_i$  are defined in (5.1).

The number of required additions is made up of three parts.

1) Additions  $A(L)$  required for the first stage  $L'$ -parallel preprocessing and post-processing matrices.

$$A(L) = \left( \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(P_{m_i \times n_i}^T) \right] + \sum_{i=1}^r \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^r m_k \right) A(Q_{m_i \times n_i}^T) \right] \right) \cdot (L/L') \quad (5.3)$$

Where  $P_{m_i \times n_i}^T$  and  $Q_{m_i \times n_i}^T$  are matrices with size  $m_i \times n_i$  and  $m_i \times n_i$  respectively.

2) Additions  $A(n)$  required for the second stage  $L''$ -parallel preprocessing and post-processing matrices

$$A(n) = \left( \sum_{i=1}^s \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^s m_k \right) A(P_{m_i \times n_i}^T) \right] + \sum_{i=1}^s \left[ \left( \prod_{j=1}^{i-1} n_j \right) \left( \prod_{k=i+1}^s m_k \right) A(Q_{m_i \times n_i}^T) \right] \right) \quad (5.4)$$

3) Additions required for the subfilters in the second stage  $L''$ -parallel FIR filter.

Therefore the total number of required additions can be given

$$A = A(L) + A(n) + \left(\frac{N}{nL'} - 1\right) \prod_{i=1}^s M_i \quad (5.5)$$

where,  $A(L)$  and  $A(n)$  are defined in (5.3) and (5.4), respectively.

The number of required delay elements is made up of four parts:

- 1) Delay elements on the input side of the first stage  $L$ -parallel FIR filter:  $(L/L)(L-1)$ ;
- 2) Delay elements on the input side of the second stage  $L$ -parallel FIR filter:  $n(n-1)$ ;
- 3) Delay elements used in the two DEMs:  $2n^2(L/L)$ ,
- 4) Delay elements required for the subfilters in the second stage  $L$ -parallel FIR filter:  $n\left(\left(\frac{N}{nL}-1\right)\prod_{i=1}^s M_i\right)$

Therefore, the total number of required delay elements is given by

$$D = (L/L)(L-1) + n(n-1) + 2n^2(L/L) + n\left(\left(\frac{N}{nL}-1\right)\prod_{i=1}^s M_i\right) \quad (5.6)$$

Note that when  $L=L$ , the direct implementation of Method-1 of 2-stage parallel FIR filter structures can be obtained.

#### 5.1.4. Example for 2-Stage Parallelism realization

Consider an example of 3-parallel 36-tap FIR filter for the implementation of 2-stage parallelism. Normally a 3-parallel 36-tap FIR filter can be implemented by ISCA as shown in figure 5.1 is given by

$$Y_3 = P_3^T H_3 Q_3^T X_3 \quad (5.7)$$

Where  $Y_3, P_3^T, H_3, Q_3^T$  and  $X_3$  are the outputs, post processing, subfilters, preprocessing and input matrices respectively as shown in figure 5.1.

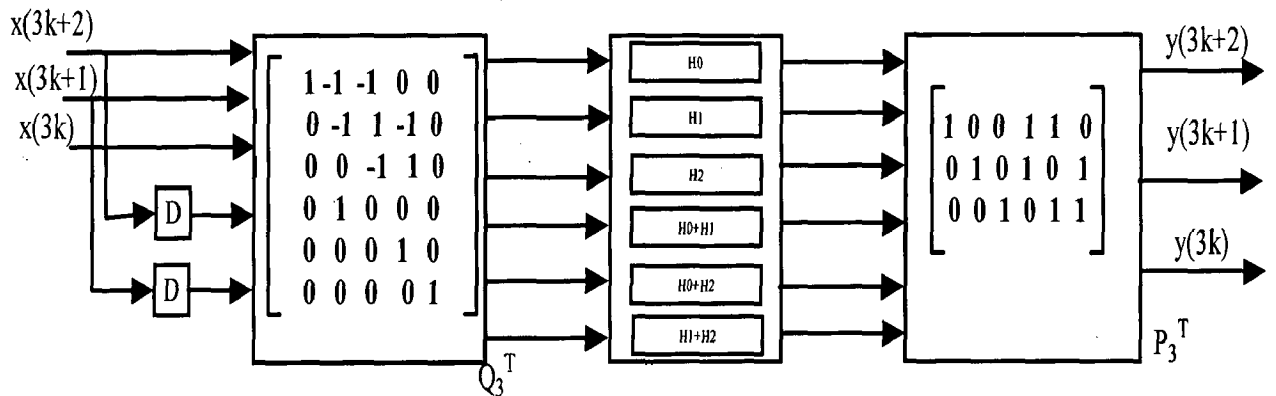


Fig 5.1 Implementation of 3-parallel FIR filter

Where  $H_0, H_1, H_2, H_0+H_1, H_0+H_2, H_1+H_2$  are 6 subfilters of length 12-tap for an 36 tap FIR filter, requiring an total of 72 multiplications. This design can process 18 input data in 6 clock cycles with all the 6 subfilters working simultaneously. Six output data of the preprocessing matrix  $Q_3^T$  are generated when 3 input data are input in each clock cycle. Each of these 6 output data of  $Q_3^T$  is processed by one of the 6 subfilters. Therefore, in 6 clock cycles, 18 data will enter  $Q_3^T$  and the 6 subfilters will process the generated 36 output data of with one subfilter processing 6 data.

The method-1 design will use the 12-tap FIR subfilter processing core to process in one clock cycle those 6 data which enter one subfilter in a row, 36 output data of  $Q_3^T$  can be done in consecutive 6 clock cycles and maintain the same processing speed. The FIR processing core, which can process 6 data in one clock cycle, is actually a 6-parallel FIR filter. The hardware cost of a 6-parallel 12-tap FIR filter is less than that of six 12-tap subfilters.. The 2-stage (method-1) parallel FIR filter structure for a 3-parallel 36-tap FIR filter is shown in figure 5.2.

The data flow of the Delay Element Matrix(DEM) in Fig. 5.3(a) is “horizontal in, vertical out” or “vertical in, horizontal out” and controlled by C0, C1, and C2 signals. C0 signal controls whether the data are “horizontal in” or “vertical in.” C1 signal controls whether the data are “horizontal out” or “vertical out.” C2 signal controls whether the data flow horizontally or vertically in the DEM. The data flow is illustrated in Figs. 5.4-5.7.

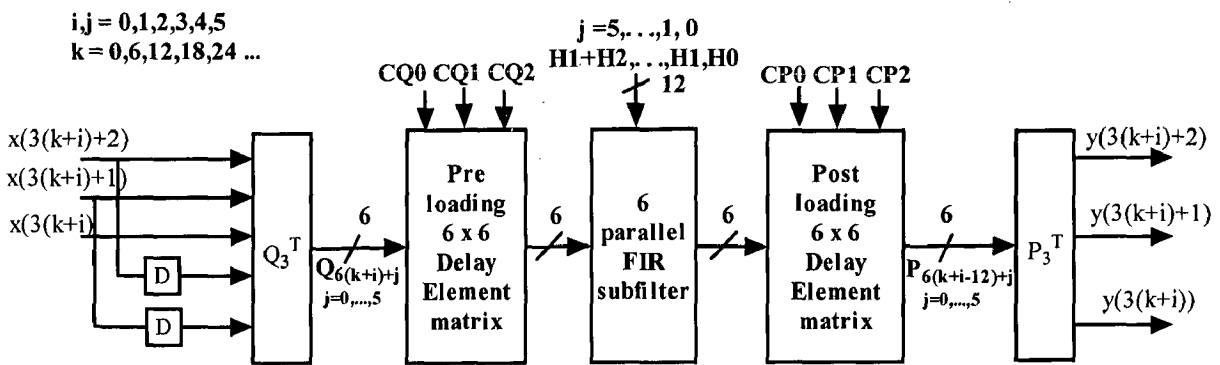


Fig 5.2 2-stage(method-1) parallelism of 3-parallel 36-tap FIR filter

In Fig. 5.4,  $Q_{ij}$ 's are the outputs of preprocessing matrix when and correspond to in Fig. 5.2. The data in preloading DEM with the same enter DEM at the same clock cycle, while those data with the same will be processed by the same subfilter. In previous parallel FIR structures, those data with the same should be each processed simultaneously by 6 independent subfilters (i.e.,  $H_0, H_1, H_2, H_0+H_1, H_0+H_2, H_1+H_2$ ). However, the 2-stage parallel FIR structures will process those data with the same by a shared filtering core in one clock cycle. Both design structures process 18 data in 6 clock cycles, leading to an effective 3-parallel processing. Fig. 5.5 shows the data flow in preloading DEM when time ranges from 5 to 11. When time is 12, the pattern of data flow will return to that of Fig. 5.4. Every 6 clock cycles, the pattern of data flow will switch between Figs. 5.4 and 5.5.

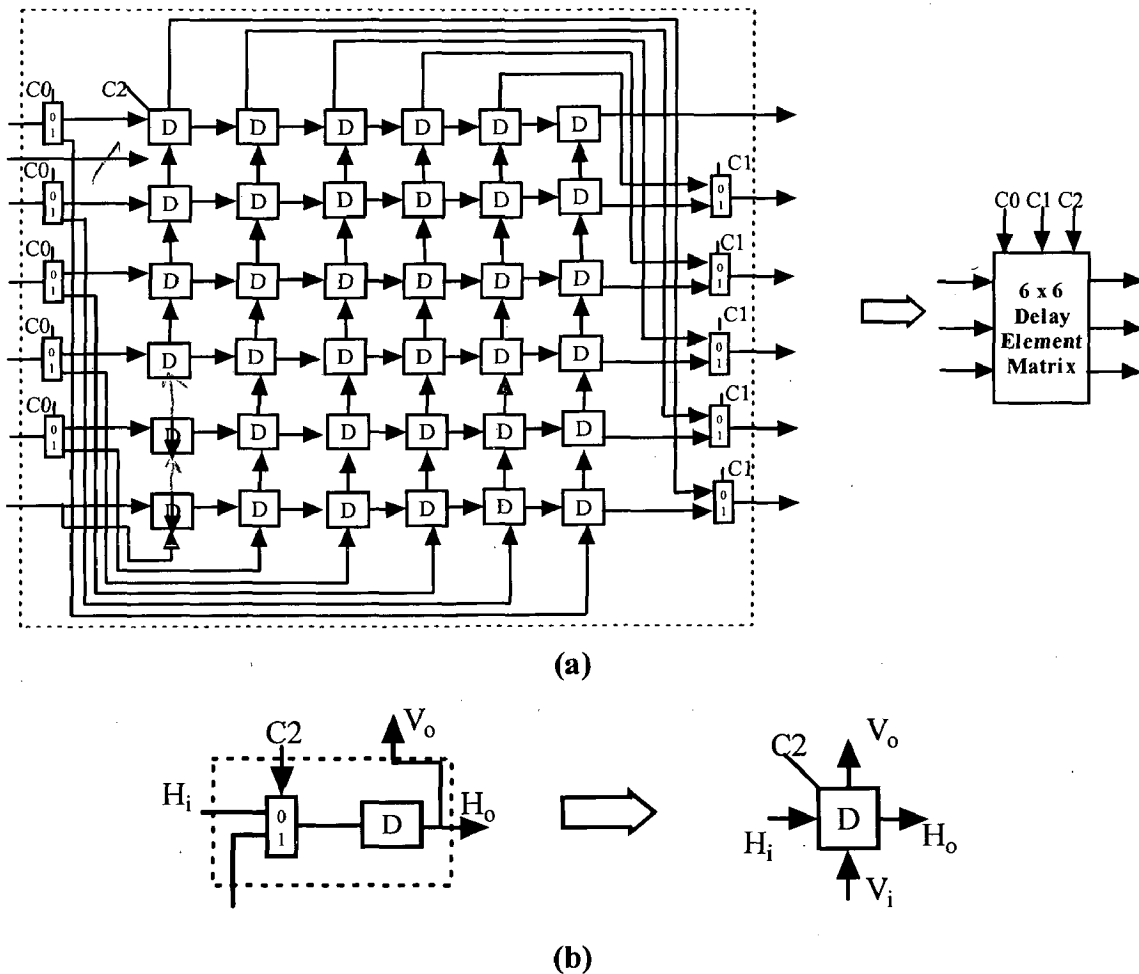


Fig 5.3 (a)  $6 \times 6$  DEM (b) Delay element function

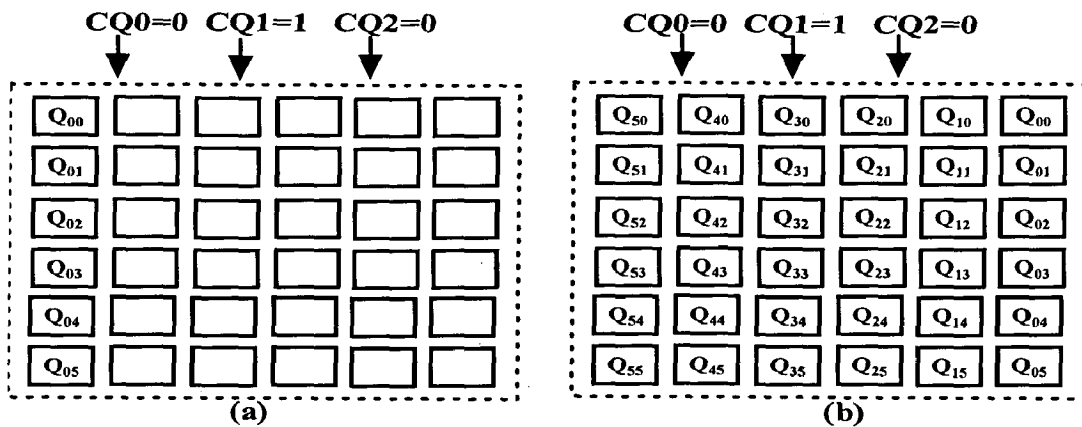


Fig 5.4 Preloading  $6 \times 6$  DEM when (a)  $k = 0$  and  $i = 0$  (i.e.,  $t = 0$ ), and  
 (b)  $k = 0$  and  $i = 5$  (i.e.,  $t = 5$ ).

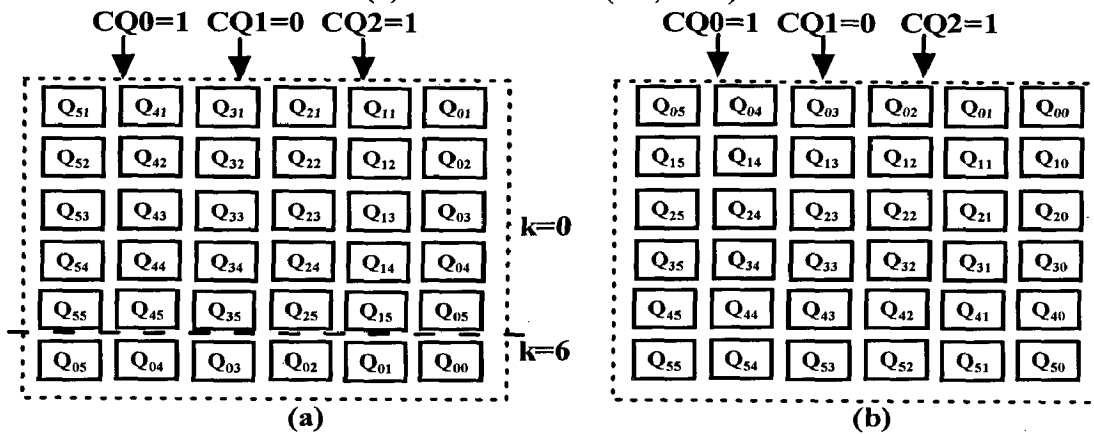


Fig 5.5 Preloading  $6 \times 6$  DEM when (a)  $k = 6$  and  $i = 0$  (i.e.,  $t = 6$ );  
 (b)  $k = 6$  and  $i = 5$  (i.e.,  $t = 11$ ).

The post-loading  $6 \times 6$  DEM works the same way as the preloading DEM. The only difference is that those  $P_{ij}$ 's in Figs.5.6 and 5.7 with the same index  $j$  enter the post-

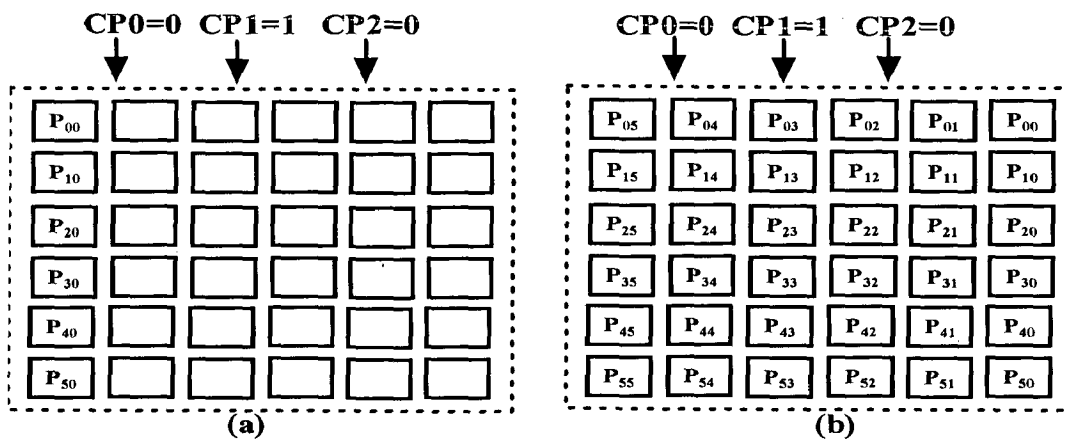


Fig 5.6 Post-loading  $6 \times 6$  DEM when (a)  $k = 12$  and  $i = 0$  (i.e.,  $t = 12$ ),  
 and (b)  $k = 12$  and  $i = 5$  (i.e.,  $t = 17$ ).

loading DEM, and those  $P_{ij}$ 's with the same index will be processed by post-processing matrix  $P_3^T$  at the same time. In Fig. 5.6(a), the first six  $P_{i0}$  ( $i=0,1,2,3,4,5$ ) enter postloading matrix when  $t=12$ , because of the latency of 12 clock cycles, which will be shown in timing analysis. Fig. 5.6 also shows the data flow in post-loading DEM when time ranges from 12 to 17. When time is 18, the pattern of data flow will switch to that of Fig. 5.7. Every 6 clock cycles, the pattern of data flow will switch between Figs. 5.6 and 5.7. The timing analysis of 3-parallel 36 tap FIR filter is as shown in fig 5.8.

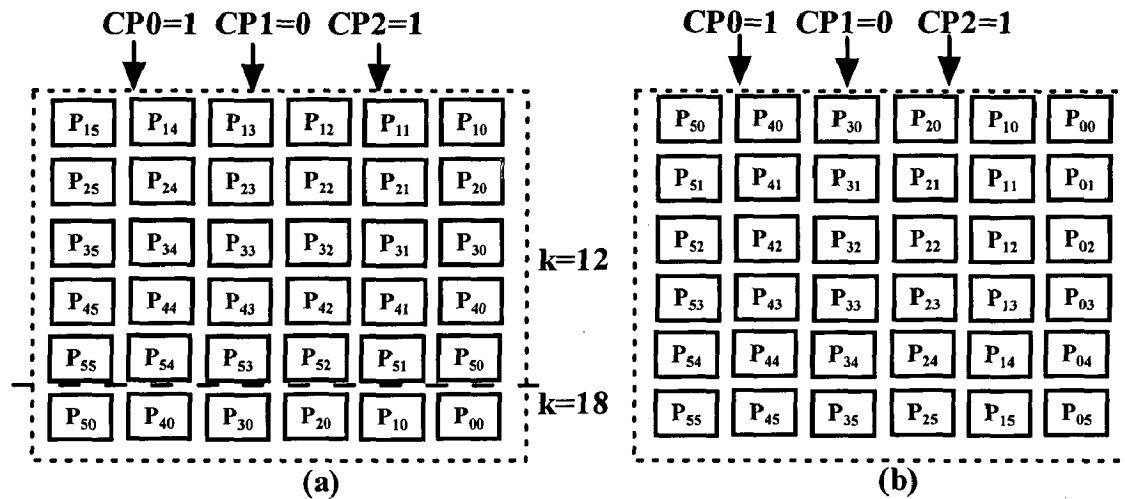


Fig 5.7 Post-loading  $6 \times 6$  DEM when (a)  $k = 18$  and  $i = 0$  (i.e.,  $t = 18$ ), and (b)  $k = 18$  and  $i = 5$  (i.e.,  $t = 23$ ).

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$Q_3^T$	Input 18 data, output 36					Input 18 data, output 36					Input 18 data, output 36					Input 18 data, output 36									
Pro-DEM	Proloading 36 data					Proloading 36 data					Proloading 36 data					Proloading 36 data									
6-parallel subfilter	Idle					Processing 36 data					Processing 36 data					Processing 36 data									
Post-DEM	Idle					Idle					Postloading 36 data					Postloading 36 data									
$P_3^T$	Idle					Idle					Input 36 data, output 18					Input 36 data, output 18									
CQ0	low					high					low					high									
CQ1	high					low					high					low									
CQ2	low					high					low					high									
CP0	unknown					unknown					low					high									
CP1	unknown					unknown					high					low									
CP2	unknown					unknown					low					high									

Fig 5.8 Timing of the 3-parallel 36-tap filter by 2-stage parallelism



### 5.1.4.1 6-Parallel FIR Subfilter as a Shared Filtering Core

An ISCA-based 6-parallel FIR filter is described by

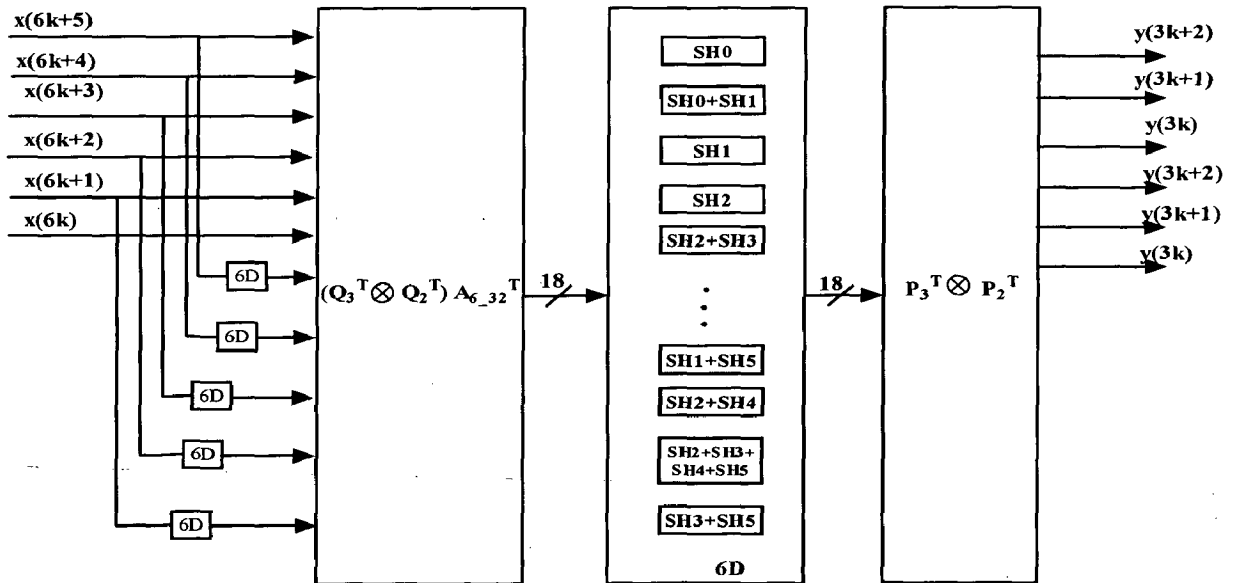
$$Y_6 = (P_3^T \otimes P_2^T) H_6 (Q_3^T \otimes Q_2^T) A_{6\_32}^T X_6 \quad (5.8)$$

Where  $P_3^T \otimes P_2^T$ ,  $Q_3^T \otimes Q_2^T$ ,  $A_{6\_32}^T$ ,  $X_6$  are obtained by ISCA as in equation (4.5) with .

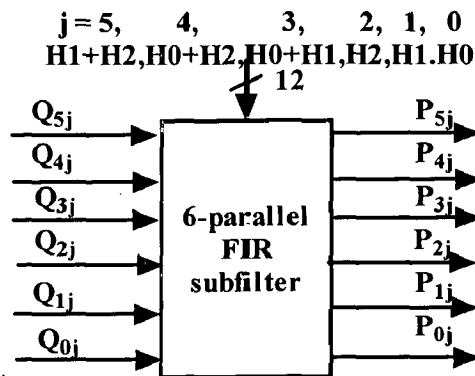
But  $H_6$  is defined as

$$H_6 = \text{diag} \left( (P_3^T \otimes P_2^T) \cdot [SH_0(j) \ SH_1(j) \ SH_2(j) \ SH_3(j) \ SH_4(j) \ SH_5(j)]^T \right)$$

represents the subfilters of the  $j^{\text{th}}$  one of the 6 subfilters  $H_0, H_1, H_2, H_0+H_1, H_0+H_2, H_1+H_2$ , which are of length 12.



(a)



(b)

Fig 5.9 (a) 6-parallel FIR filter as shared filtering core (b) block diagram of (a)

VLSI structure of a 6-parallel FIR subfilter as a shared filtering core is shown as in Fig. 5.9. This 6-parallel FIR subfilter is derived from the ISCA-based 6-parallel FIR filter by replacing the delay element “ $D$ ” with “ $6D$ ” which is because this 6-parallel FIR subfilter will be shared by the 6 subfilters  $H_0, H_1, H_2, H_0+H_1, H_0+H_2, H_1+H_2$ . Therefore, the hardware cost of this 6-parallel FIR subfilter is the same as that of the ISCA-based 6-parallel FIR filter except the 6-fold increase in the number of the delay elements. The total number of required multiplications, additions and delay elements of the 6-parallel 12-tap FIR subfilter are 36, 70 and 138 respectively. The preprocessing and post-processing require 52 additions. The subfilter length of this 6-parallel 12-tap FIR subfilter is  $12/6=2$ . From Figs. 5.1 and 5.2, the computation process of preprocessing and post-processing of the ISCA and the 2-stage design are exactly the same and the differences are located in the subfilter part. The 2-stage 3-parallel 36-tap FIR filter can save 36 multiplications at the cost of 4 additions and 144 delay elements.

#### 5.1.4.2 Method-2 realization:

In method-2, the increase  $n$  will be controlled, when  $L$  is large. The first stage 3-parallel FIR filter has 6 subfilters ( $n=6$ ). If 36 data can be processed in 6 clock cycles, an equivalent 6-parallel implementation can be obtained. When  $L=3$ , 36 data will generate 72 output data of  $Q_3^T$  and the 6 subfilters of length  $36/3=12$  will process the generated 72 output data of with one subfilter processing 12 data. 12 data will be processed by one of the 6 subfilters in one clock cycle. When a shared filtering core is designed, a 12-parallel FIR subfilter of length  $12/12=1$  is used as shown in figure 4.10. As shown in the above analysis, the 36 data must enter the  $6 \times 6$  preloading DEM in 6 clock cycles. But one  $Q_3^T$  can only process 18 inputs in 6 clock cycles. Therefore, two  $Q_3^T$  and two  $6 \times 6$  preloading DEM's are used on the input side in Fig.4.10. Meanwhile, two  $P_3^T$  and two  $6 \times 6$  post-loading DEM's are used on the output side for the same reason. The second stage 12-parallel FIR subfilter module requires 66 delay elements on its input side. Since its 54 subfilters are all 1-tap, the number of required multiplications, additions and delay elements for subfilters are  $54 \times 1=54$ ,  $54 \times 0=0$  and  $54 \times 0=0$ . The first stage preprocessing and post-processing matrices require 24 additions. The second stage preprocessing and

postprocessing matrices require 192 additions. Thus the number of required multiplications, additions and delay elements for the 6-parallel 36-tap FIR filter are 54,  $192+24=216$  and  $36 \times 4 + 11 \times 6 + 4 = 214$  respectively. The timing analysis of 6-parallel 36-tap FIR filter is shown in figure 5.11.

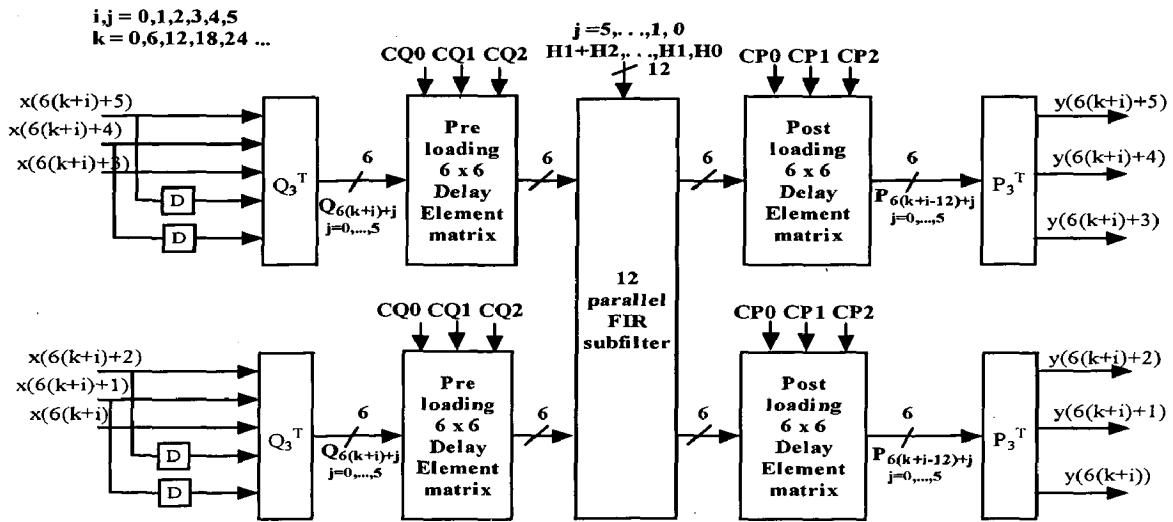


Fig 5.10 2-stage parallel FIR filter for an 6-parallel 36 tap FIR filter

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$Q_3^T$	Input 36 data, output 72						Input 36 data, output 72						Input 36 data, output 72						Input 36 data, output 72						
Pro-DEM	Proloading 72 data						Proloading 72 data						Proloading 72 data						Proloading 72 data						
12-parallel subfilter	Idle						Processing 72 data						Processing 72 data						Processing 72 data						
Post-DEM	Idle						Idle						Postloading 72 data						Postloading 72 data						
$P_3^T$	Idle						Idle						Input 72 data, output 36						Input 72 data, output 36						
CQ0	low						high						low						high						
CQ1	high						low						high						low						
CQ2	low						high						low						high						
CP0	unknown						unknown						low						high						
CP1	unknown						unknown						high						low						
CP2	unknown						unknown						low						high						

Fig 5.11 Timing of 6-parallel 36-tap FIR filter

## 6 Application of Parallel FIR filters

The main applications [4] of parallel FIR filters are in equalizers, 2D parallel FIR filters, 2D Discrete Wavelet Transform(DWT) . In this chapter, the high speed implementation of 2D DWT based on hardware efficient parallel FIR filters are realized.

### 6.1 2D Discrete Wavelet Transform

The two-dimensional (2-D) discrete wavelet transform (DWT) [12]–[14] is a mathematical technique that decomposes a 2-D discrete signal in a multiresolution space domain by using dilated/contracted and translated versions of a single finite duration basis function, named the prototype wavelet. The discrete wavelet transform (DWT) has been widely used in audio and image processing, digital communications and other application fields. This computation transform has been widely implemented in very-large-scale integration (VLSI) [12]–[16][21][22][23] because of the real-time requirement. DWT has traditionally been implemented by convolution or FIR filter bank-based structures [12]–[16][21][22][23]. At present, many VLSI architectures for the 2-D DWT have been proposed to meet the requirements of real-time processing. However, because the filtering operations are required in both the horizontal and vertical directions, designing a highly efficient architecture at a low cost is difficult.

Fast algorithm based parallel FIR filter structures are designed to improve the processing speed and control the increase of the hardware cost at the same time. This design can reduce the computation time of the reported fastest 2-D DWT architectures [12] with filter length 4 from  $\frac{N^2}{3}$  to  $\frac{N^2}{12}$ , but the number of required multipliers is only 3 times that of [12]. Higher processing speed can be achieved when parallel FIR structures with higher parallelism levels [17] are used. Furthermore, since the filtering structures are regular, the control signals are very simple.

#### 6.1.1 2D Non separable DWT structure based on parallel FIR filters

The 2-D DWT consists of computing the 1-D DWT of each of the  $N$  rows of the original  $N \times N$  image and then computing the 1-D DWT of each of the resulting  $N$  columns [22].

An efficient 1-D DWT decimation filter has been described in [12] and [13]. Although this decimation filter can save the number of both multipliers by a half, it has two drawbacks: 1) the input sampling frequency must be two times as fast as the output frequency, in order to get an output at every clock cycle; 2) this decimation filter cannot be easily operated at a higher processing speed.

The design is illustrated through an example. Let us take an image of size  $8 \times 8$  as in (6.1),

$$X = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} & x_{07} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\ x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\ x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\ x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\ x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \end{bmatrix} \quad (6.1)$$

filter length as 4, the low and high pass filter coefficients as  $H = \{a, b, c, d\}$  and  $G = \{e, f, g, h\}$  respectively. First apply low-pass 1-DDWT to (6.1) in the row dimension. After low-pass filtering and down sampling by 2, (6.2) can be obtained from (6.1). Equation (6.2) can be simplified as (6.3), where  $i = 0, 1, \dots, 7$  and it can be transformed into matrix form and represented as (6.4). From (6.4), 1-D DWT in (6.3) has been transformed into two FIR filters each with a filter length of half of the original filter. A low pass 2-parallel 2-tap FIR filter can be get from ISCA with  $H_2$

as  $H_2 = \text{diag}[b \quad b+d \quad d]$ ,  $Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$  and the remaining  $P_2, X_2, Y_2$  are same as

that equation (4.7) as shown in figure 6.1. Similarly a high pass filter can be obtained by replacing  $b, d$  and  $a, c$  with  $f, h$  and  $e, g$  respectively.

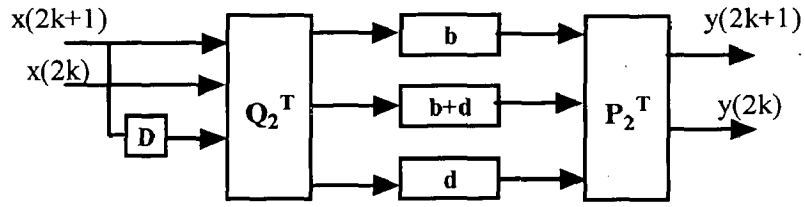


Fig 6.1 2-parallel 2-tap FIR filter

$$\begin{aligned}
 m &= \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} \begin{bmatrix} b & a & 0 & 0 & 0 & 0 & 0 & 0 \\ d & c & b & a & 0 & 0 & 0 & 0 \\ 0 & 0 & d & c & b & a & 0 & 0 \\ 0 & 0 & 0 & 0 & d & c & b & a \end{bmatrix}^T \\
 &= \begin{bmatrix} bx_{00} + ax_{01} & dx_{00} + cx_{01} + bx_{02} + ax_{03} & dx_{02} + cx_{03} + bx_{04} + ax_{05} & dx_{04} + cx_{05} + bx_{06} + ax_{07} \\ bx_{10} + ax_{11} & dx_{10} + cx_{11} + bx_{12} + ax_{13} & dx_{12} + cx_{13} + bx_{14} + ax_{15} & dx_{14} + cx_{15} + bx_{16} + ax_{17} \\ bx_{20} + ax_{21} & dx_{20} + cx_{21} + bx_{22} + ax_{23} & dx_{22} + cx_{23} + bx_{24} + ax_{25} & dx_{24} + cx_{25} + bx_{26} + ax_{27} \\ bx_{30} + ax_{31} & dx_{30} + cx_{31} + bx_{32} + ax_{33} & dx_{32} + cx_{33} + bx_{34} + ax_{35} & dx_{34} + cx_{35} + bx_{36} + ax_{37} \\ bx_{40} + ax_{41} & dx_{40} + cx_{41} + bx_{42} + ax_{43} & dx_{42} + cx_{43} + bx_{44} + ax_{45} & dx_{44} + cx_{45} + bx_{46} + ax_{47} \\ bx_{50} + ax_{51} & dx_{50} + cx_{51} + bx_{52} + ax_{53} & dx_{52} + cx_{53} + bx_{54} + ax_{55} & dx_{54} + cx_{55} + bx_{56} + ax_{57} \\ bx_{60} + ax_{61} & dx_{60} + cx_{61} + bx_{62} + ax_{63} & dx_{62} + cx_{63} + bx_{64} + ax_{65} & dx_{64} + cx_{65} + bx_{66} + ax_{67} \\ bx_{70} + ax_{71} & dx_{70} + cx_{71} + bx_{72} + ax_{73} & dx_{72} + cx_{73} + bx_{74} + ax_{75} & dx_{74} + cx_{75} + bx_{76} + ax_{77} \end{bmatrix} \\
 &= \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \\ m_{40} & m_{41} & m_{42} & m_{43} \\ m_{50} & m_{51} & m_{52} & m_{53} \\ m_{60} & m_{61} & m_{62} & m_{63} \\ m_{70} & m_{71} & m_{72} & m_{73} \end{bmatrix} \tag{6.2}
 \end{aligned}$$

$$[m_{70} \quad m_{71} \quad m_{72} \quad m_{73}] = [bx_{70} + ax_{71} \quad dx_{70} + cx_{71} + bx_{72} + ax_{73} \quad dx_{72} + cx_{73} + bx_{74} + ax_{75} \quad dx_{74} + cx_{75} + bx_{76} + ax_{77}] \tag{6.3}$$

$$\begin{bmatrix} m_{i0} \\ m_{i1} \\ m_{i2} \\ m_{i3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & x_{i0} & x_{i1} \\ x_{i0} & x_{i1} & x_{i2} & x_{i3} \\ x_{i2} & x_{i3} & x_{i4} & x_{i5} \\ x_{i4} & x_{i5} & x_{i6} & x_{i7} \end{bmatrix} \begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} 0 & x_{i0} \\ x_{i0} & x_{i2} \\ x_{i2} & x_{i4} \\ x_{i4} & x_{i6} \end{bmatrix} \begin{bmatrix} d \\ b \end{bmatrix} + \begin{bmatrix} 0 & x_{i1} \\ x_{i1} & x_{i3} \\ x_{i3} & x_{i5} \\ x_{i5} & x_{i7} \end{bmatrix} \begin{bmatrix} c \\ a \end{bmatrix} \quad (6.4)$$

If 2-parallel FIR filter structure is applied to (6.4), the computation of  $[m_{i0} \ m_{i1} \ m_{i2} \ m_{i3}]$  requires just two clock cycles, with  $[m_{i0} \ m_{i1}]$  coming out first and then  $[m_{i2} \ m_{i3}]$  coming out after two clock cycles; the row filtering of (6.1) can be completed by the architecture shown in Fig.6.2. Only low pass section is shown, the high pass section is similar to it having the same input data, can be obtained by just replacing  $b, d$  and  $a, c$  with  $f, h$  and  $e, g$  respectively.

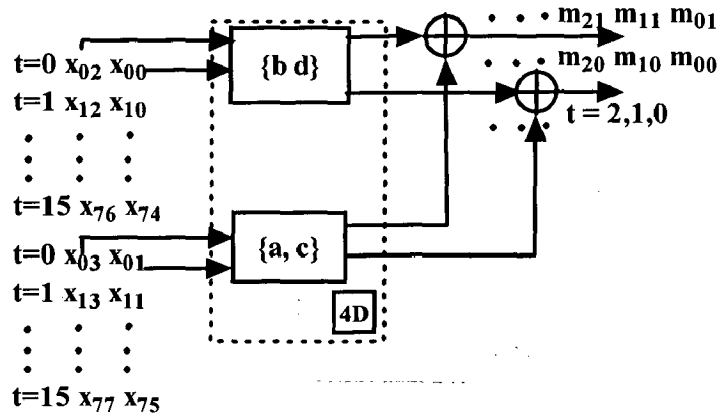


Fig 6.2 1D DWT (i.e., after row filtering) based on 2-parallel FIR filter

Note that "4D" in Fig. 6.2 is used to replace every "D" in the original parallel FIR filter shown in Fig.6.1 because the parallel filter is shared by four rows of input data, which are processed as shown in Fig. 6.2. After filtering the first through fourth columns of the first through fourth rows of (6.1), filter the fifth through eighth columns of the first through fourth rows of (6.1). After filtering the first through fourth rows is finished, filter the first through fourth rows starting from the first through fourth columns. Now apply high-pass 1-D DWT to (6.2) in column dimension. After high-pass filtering and down sampling by 2, we can get (6.5) from (6.2). If low-pass 1-D DWT is applied to (6.2) in column dimension, after low-pass filtering and down sampling by 2, equation (6.6) can be obtained from (6.2). It is obvious that the computations of the first row of (6.5) and of

(6.6) require that of the first and second rows of (6.2); the second rows of (6.5) and (6.6) require that of the third and fourth rows and previously computed first and second rows of (6.1); the computation of the  $i^{\text{th}}$  row of (6.9) and (6.10) requires that of  $(2i-1)^{\text{th}}$  and  $(2i)^{\text{th}}$  and previously computed  $(2i-3)^{\text{rd}}$  and  $(2i-2)^{\text{nd}}$  rows of (6.6). Equation (6.5) can be simplified as equation (6.7) and equation (6.6) can be simplified as equation (6.8).

$$\begin{aligned}
 HG &= \begin{bmatrix} fm_{00} + em_{10} & fm_{01} + em_{11} & fm_{02} + em_{12} & fm_{03} + em_{13} \\ hm_{00} + gm_{10} + fm_{20} + em_{30} & hm_{01} + gm_{11} + fm_{21} + em_{31} & hm_{02} + gm_{12} + fm_{22} + em_{32} & hm_{03} + gm_{13} + fm_{23} + em_{33} \\ hm_{20} + gm_{30} + fm_{40} + em_{50} & hm_{21} + gm_{31} + fm_{41} + em_{51} & hm_{22} + gm_{32} + fm_{42} + em_{52} & hm_{23} + gm_{33} + fm_{43} + em_{53} \\ hm_{40} + gm_{50} + fm_{60} + em_{70} & hm_{41} + gm_{51} + fm_{61} + em_{71} & hm_{42} + gm_{52} + fm_{62} + em_{72} & hm_{43} + gm_{53} + fm_{63} + em_{73} \end{bmatrix} \\
 &= \begin{bmatrix} HG_{00} & HG_{01} & HG_{02} & HG_{03} \\ HG_{10} & HG_{11} & HG_{12} & HG_{13} \\ HG_{20} & HG_{21} & HG_{22} & HG_{23} \\ HG_{30} & HG_{31} & HG_{32} & HG_{33} \end{bmatrix}
 \end{aligned} \tag{6.5}$$

$$\begin{aligned}
 HH &= \begin{bmatrix} bm_{00} + am_{10} & bm_{01} + am_{11} & bm_{02} + am_{12} & bm_{03} + am_{13} \\ dm_{00} + cm_{10} + bm_{20} + am_{30} & dm_{01} + cm_{11} + bm_{21} + am_{31} & dm_{02} + cm_{12} + bm_{22} + am_{32} & dm_{03} + cm_{13} + bm_{23} + am_{33} \\ dm_{20} + cm_{30} + bm_{40} + am_{50} & dm_{21} + cm_{31} + bm_{41} + am_{51} & dm_{22} + cm_{32} + bm_{42} + am_{52} & dm_{23} + cm_{33} + bm_{43} + am_{53} \\ dm_{40} + cm_{50} + bm_{60} + am_{70} & dm_{41} + cm_{51} + bm_{61} + am_{71} & dm_{42} + cm_{52} + bm_{62} + am_{72} & dm_{43} + cm_{53} + bm_{63} + am_{73} \end{bmatrix} \\
 &= \begin{bmatrix} HH_{00} & HH_{01} & HH_{02} & HH_{03} \\ HH_{10} & HH_{11} & HH_{12} & HH_{13} \\ HH_{20} & HH_{21} & HH_{22} & HH_{23} \\ HH_{30} & HH_{31} & HH_{32} & HH_{33} \end{bmatrix}
 \end{aligned} \tag{6.6}$$

$$\begin{bmatrix} HG_{0j} \\ HG_{1j} \\ HG_{2j} \\ HG_{3j} \end{bmatrix} = \begin{bmatrix} fm_{0j} + em_{1j} \\ hm_{0j} + gm_{1j} + fm_{2j} + em_{3j} \\ hm_{2j} + gm_{3j} + fm_{4j} + em_{5j} \\ hm_{4j} + gm_{5j} + fm_{6j} + em_{7j} \end{bmatrix} = \begin{bmatrix} m_{0j} & 0 \\ m_{2j} & m_{0j} \\ m_{4j} & m_{2j} \\ m_{6j} & m_{4j} \end{bmatrix} \begin{bmatrix} f \\ h \end{bmatrix} + \begin{bmatrix} m_{1j} & 0 \\ m_{3j} & m_{1j} \\ m_{5j} & m_{3j} \\ m_{7j} & m_{5j} \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} \tag{6.7}$$

$$\begin{bmatrix} HH_{0j} \\ HH_{1j} \\ HH_{2j} \\ HH_{3j} \end{bmatrix} = \begin{bmatrix} bm_{0j} + am_{1j} \\ dm_{0j} + cm_{1j} + bm_{2j} + am_{3j} \\ dm_{2j} + cm_{3j} + bm_{4j} + am_{5j} \\ dm_{4j} + cm_{5j} + bm_{6j} + am_{7j} \end{bmatrix} = \begin{bmatrix} m_{0j} & 0 \\ m_{2j} & m_{0j} \\ m_{4j} & m_{2j} \\ m_{6j} & m_{4j} \end{bmatrix} \begin{bmatrix} b \\ d \end{bmatrix} + \begin{bmatrix} m_{1j} & 0 \\ m_{3j} & m_{1j} \\ m_{5j} & m_{3j} \\ m_{7j} & m_{5j} \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} \tag{6.8}$$

From (6.7) and (6.8), the computations of (6.5) and (6.6) have each been transformed into two FIR filters each with a filter length of half of the original filter. If two 2-parallel FIR filter are used as shown in Fig. 6.1, two column elements in each row of (6.2) is



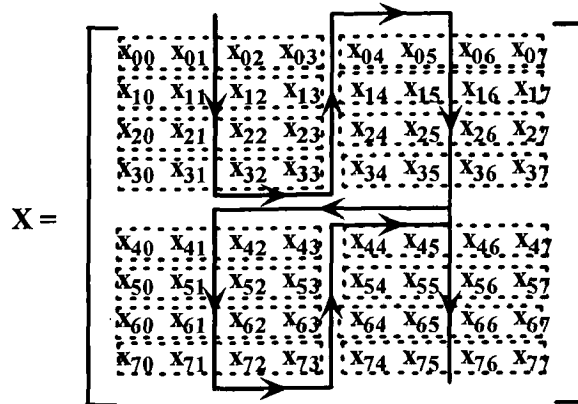


Fig 6.3 Input data flow of 2D DWT of 2-parallel FIR filter

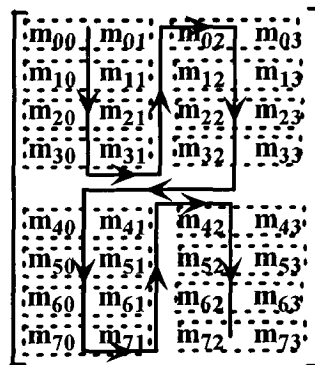
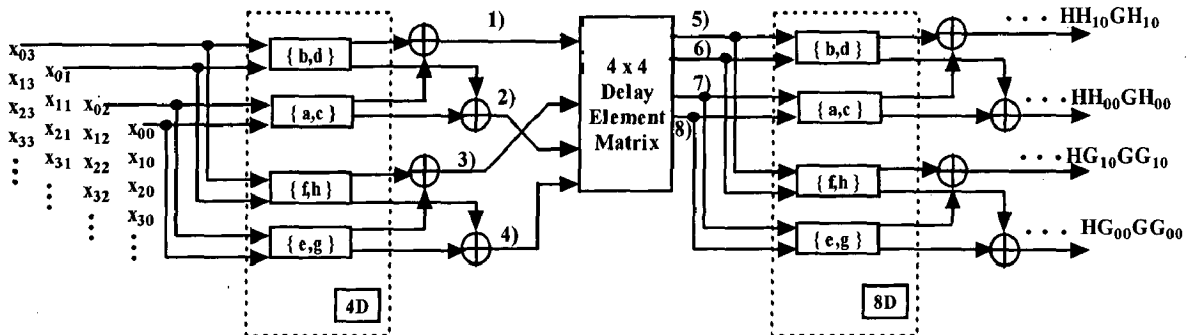


Fig 6.4 Output data flow of 1D DWT of 2-parallel FIR filter

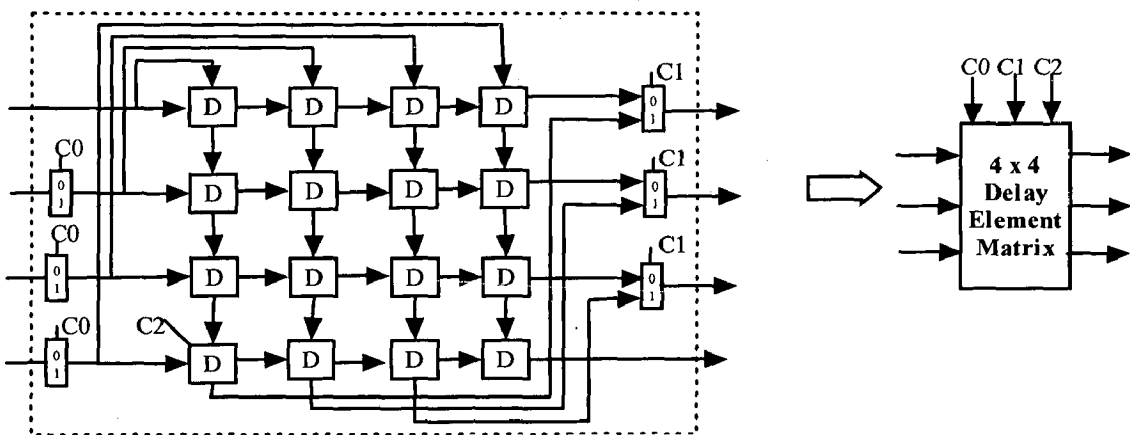
computed in each clock cycle, and then if two 2-parallel FIR filters are used with  $m_j$  as input, two row elements in each column of (6.7) or (6.8) can be computed in one clock cycle. The only problem is that the computation of (6.2) with two 2-parallel FIR filters outputs two data  $[m_{i0} \ m_{i1}]$  or  $[m_{i2} \ m_{i3}]$  in one clock cycle, which has been shown in Fig. 6.4, but that of (6.7) or (6.8) with two 2-parallel FIR filters requires four data  $[m_{0j} \ m_{1j} \ m_{2j} \ m_{3j}]$  as input in each clock cycle. A  $4 \times 4$  DEM similar to Fig. 6.5(b) is used to solve this problem. The 2-D non-separable DWT structure for image size of  $8 \times 8$ , which computes the first resolution level of an  $N \times N$  image in  $N^2/4$  clock cycles, is shown in Fig. 6.5. The delay elements in the  $4 \times 4$  DEM have the functionality as shown in Fig. 6.5. From Fig. 6.5, the 2-D non-separable DWT can finish computing the first-resolution-level  $HG, HH, GG$  and  $GH$  of an  $8 \times 8$  image in 16 clock cycles. Output data flow of delay element matrix is shown in Fig. 6.6.

Note that the two input data to the 2-parallel filter  $\{a,c\}$  are the same as those to  $\{e,g\}$ ; thus, these two 2-parallel filters can share the same  $Q_2^T$  block and delay element of the input side as shown in Fig. 6.1 and four addition operations can be saved. Another four adders can be saved from sharing the same input data to the two 2-parallel filters  $\{b,d\}$  and  $\{f,h\}$ . This sharing can also lead to the saving of  $4 \times 2 = 8$  delay elements of the first level and  $8 \times 2 = 16$  delay elements of the second level. The total hardware cost of this architecture is 24 multiplications, 32 additions, and 40 delay elements. In general, the computation time for the first-resolution-level 2-D DWT of an  $N \times N$  image is  $N^2/4$  clock cycles by using 2-parallel FIR filters.



- 1)...  $m_{31} m_{21} m_{11} m_{01}$     3)...  $m'_{31} m'_{21} m'_{11} m'_{01}$     5)...  $m_{31} m'_{31} m_{30} m'_{30}$     7)...  $m_{21} m'_{21} m_{20} m'_{20}$   
 2)...  $m_{30} m_{20} m_{10} m_{00}$     4)...  $m'_{30} m'_{20} m'_{10} m'_{00}$     6)...  $m_{11} m'_{11} m_{10} m'_{10}$     8)...  $m_{01} m'_{01} m_{00} m'_{00}$

(a)



(b)

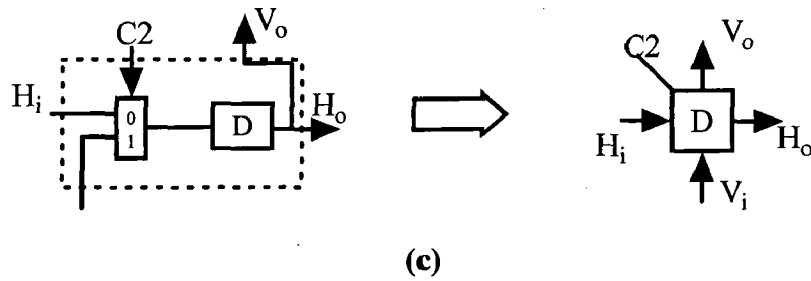


Figure 6.5 (a)  $N^2/4$  2D DWT structure for an  $8 \times 8$  image (b)  $4 \times 4$  Delay Element Matrix  
(c) Delay Element function

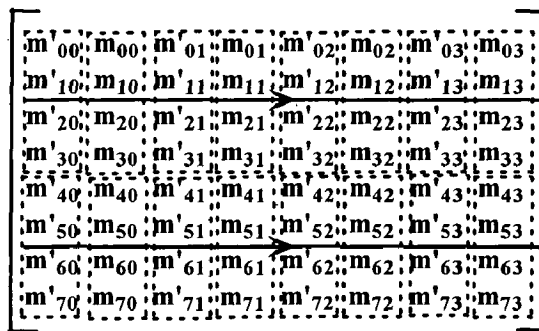


Fig 6.6 Output data flow of  $4 \times 4$  DEM

### 6.2.2 2-D DWT of an $N \times N$ Image by $L$ - Parallel FIR Filtering

Now, the algorithm for computing the first level 2-D DWT of an  $N \times N$  image by applying  $L$ -level parallel FIR filtering can be generalized. For a 2-D DWT with low-pass filter  $H$  and high-pass filter  $G$ , first decimate  $H$  and  $G$  by factor 2 into  $\{H_0, H_1\}$  and  $\{G_0, G_1\}$ .  $H_0, H_1, G_0, G_1$  are all subfilters. The 2-D nonseparable DWT structure, which computes the first-resolution-level of an  $N \times N$  image in  $N^2/4$  clock cycles, is shown in Fig. 6.7. From Fig. 6.7, the subfilters  $H_0$  and  $G_0$  share the same input data for both the first and the second level of computation. Subfilters  $H_1$  and  $G_1$  also share the same data for both levels of computation. This property can save large number of delay elements especially for the second level of computation, because the saving of each delay element in the second level filter structure will lead to the saving of  $N$  delay elements of the final 2-DDWT structure. When parallelism level  $L$  is greater than 1, adders can also be saved.

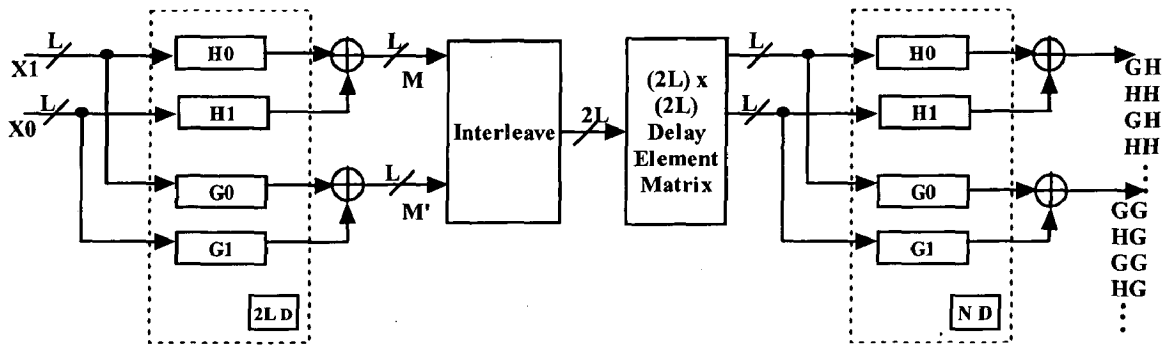


Fig 6.7  $N^2/2L$  2D DWT structure for an  $N \times N$  image

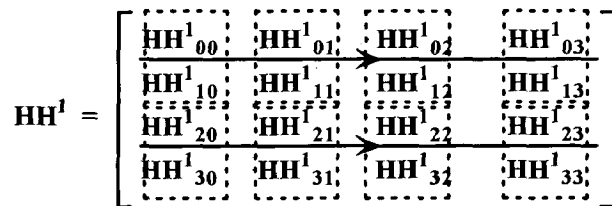


Fig 6.8 Output data flow of the first level  $N^2/4$  2D DWT structure

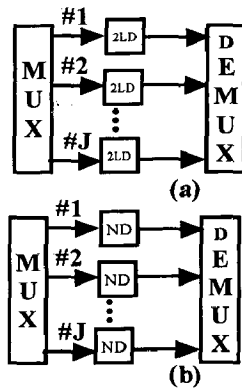


Fig 6.9 Interleaving structures of 2D DWT structures for an  $N \times N$  image

(a) row interleave (b) column interleave

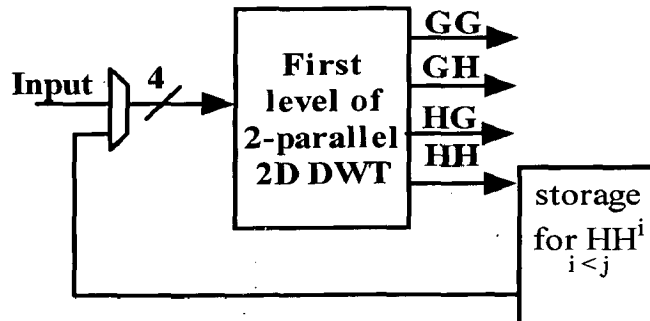


Fig 6.10 Hardware implementation of 2D DWT for an image of

$N \times N$  size with J-level resolution in  $N^2/3$  clock cycles.

For an  $8 \times 8$  image in (6.1), the hardware implementation of its 2-D DWT with  $J$ -level resolution and a computation time of  $N^2/3$  is shown in Fig. 6.10. The 2-D DWT structure in Fig. 6.7 works as follows. The outputs  $HH_j^1$  of the first resolution level  $N^2/4$ DWT structure are also the inputs of the second resolution level  $N^2/4$  DWT structure. As shown in Fig. 6.5,  $HH_j^1$  from a  $N^2/4$ DWT structure are generated in the sequence shown in Fig. 6.8. Comparing the data flow in Fig. 6.8 with the input data flow of a  $N^2/4$  DWT structure as shown in Fig. 6.3,  $N+8$  storage elements are needed for the output of the first resolution level  $HH$  before starting the computation of the second resolution level 2-D DWT, and  $N/2+8$  storage elements are needed for the output of the second level  $HH$  before the start of the computation of the third level 2-D DWT, and so on. Thus, the total storage elements for  $HH^i$ ,  $i < j$  is given by

$$(N+8)+(N/2+8)+(N/4+8)+\dots = 2N+8(J-1) \quad (6.13)$$

Since the computation of different resolution levels of 2-D DWT are shared, the computation of  $HH^i$  will be interrupted whenever higher level  $HH^{i+1}$  is ready to be fetched from the storage and computed. In order to resume the computation of lower level  $HH^i$ , we need to save the intermediate computation results for  $HH^i$  and interleaving filtering is used. To interleave the filtering of the first-resolution-level  $N^2/4$ DWT structure shown in Fig. 6.5, replace “4D” and “8D” in Fig. 6.5 with Fig. 6.9(a) and (b), respectively [4]. In Fig. 6.9, every pair of DEMUX and MUX switch to next channel  $i$  when the  $i^{th}$  level of 2-D DWT is being computed.

The data flow of the hardware implementation of the 2-D DWT of an  $8 \times 8$  image with two-level resolution and a computation time of  $N^2/3$  is shown in Table 6.1. In Table 6.1,  $x_{ij}$  and  $y_{ij}$  are from two different  $8 \times 8$  images. Let us assume that these two images are processed in a row.  $HH_{00}^{1x}$  and  $HH_{00}^{2x}$  are the outputs of the first and second resolution levels of the image represented as  $x_{ij}$ . From Table 6.1, we can see that there is a latency of 4 clock cycles between the input of  $HH_{00}^{1x}$  and the output of  $HH_{00}^{2x}$  because of the  $4 \times 4$  delay element matrix in the first level DWT structure as shown in Fig. 6.5. From

Table 6.1, we can also see that input data of the  $i^{\text{th}}$  resolution level will be interrupted by those of the  $(i+1)^{\text{th}}$  resolution level,  $(N/2^{i-1} - (2L)^2)$  clock cycles after the first input data of the  $i^{\text{th}}$  resolution level have been given. Only after the available data of resolution level higher than  $i$  are processed, the available data of resolution level  $i$  can be processed.

### 6.2.3 Computational Complexity

The hardware cost of the 2D DWT structures of an  $N \times N$  image in terms of the number of required multipliers (R.M.), adders (R.A.), and delay elements (R.D.) with resolution level  $J$  as

$$\begin{aligned}
 R.M. &= 2 \cdot \sum_{i=0}^1 (M(H_i, L) + M(G_i, L)) \\
 R.A. &= 2 \cdot \sum_{i=0}^1 (A(H_i, L) + A(G_i, L) - A(Q_{H_i}^T, L) - A(Q_{G_i}^T, L)) + 4 \cdot L \\
 R.D. &= [(2 \cdot L) \cdot J + 2N] + \sum_{i=0}^1 (D(H_i, L) + D(G_i, L)) + 4 \cdot L^2
 \end{aligned} \tag{6.10}$$

Where  $M(H_i, L)$ ,  $A(H_i, L)$  and  $D(H_i, L)$  are the number of required multipliers, adders and delay elements for implementation of a  $L$ -parallel subfilter  $H_i$ , respectively.  $A(Q_{H_i}^T, L)$  is the number of required adders for the  $Q_{H_i}^T$  block of a  $L$ -parallel subfilter  $H_i$ .

The computation time is

$$\frac{N^2}{2L} \left( 1 + \frac{1}{4} + \frac{1}{16} + \dots \right) = \frac{2N^2}{3L}$$

For  $L=2$ , the computation time is  $N^2/3$  as shown in table 6.1.

A 4-parallel 2D DWT structure can be obtained by keeping  $L=4$  in figure 6.7. The interleaving structures will have a delay of 8D at both the row and column 1D DWT. This structure will have an  $8 \times 8$  DEM for the transition of data from row to column process of DWT. The data flow is similar to the 2-parallel 2D DWT. The structure of 4-parallel 2-tap FIR filter is in [27]. The computation time of 4-parallel 2D DWT is  $N^2/6$ .

Table 6.1

Data flow of 2-D DWT of a  $8 \times 8$  image with 2-level resolution  
and a computation time of  $N^2/3$  clock cycles

t	HH	HG	GH	GG	Input
0	-	-	-	-	$x_{00}, x_{01}, x_{02}, x_{03}$
1	-	-	-	-	$x_{10}, x_{11}, x_{12}, x_{13}$
2	-	-	-	-	$x_{20}, x_{21}, x_{22}, x_{23}$
3	-	-	-	-	$x_{30}, x_{31}, x_{32}, x_{33}$
4	-	-	$GH_{00}^{1x} GH_{10}^{1x}$	$GG_{00}^{1x} GG_{10}^{1x}$	$x_{04}, x_{05}, x_{06}, x_{07}$
5	$HH_{00}^{1x} HH_{10}^{1x}$	$HG_{00}^{1x} HG_{10}^{1x}$	-	-	$x_{14}, x_{15}, x_{16}, x_{17}$
6	-	-	$GH_{01}^{1x} GH_{11}^{1x}$	$GG_{01}^{1x} GG_{11}^{1x}$	$x_{24}, x_{25}, x_{26}, x_{27}$
7	$HH_{01}^{1x} HH_{11}^{1x}$	$HG_{01}^{1x} HG_{11}^{1x}$	-	-	$x_{34}, x_{35}, x_{36}, x_{37}$
8	-	-	$GH_{02}^{1x} GH_{12}^{1x}$	$GG_{02}^{1x} GG_{12}^{1x}$	$x_{40}, x_{41}, x_{42}, x_{43}$
9	$HH_{02}^{1x} HH_{12}^{1x}$	$HG_{02}^{1x} HG_{12}^{1x}$	-	-	$x_{50}, x_{51}, x_{52}, x_{53}$
10	-	-	$GH_{03}^{1x} GH_{13}^{1x}$	$GG_{03}^{1x} GG_{13}^{1x}$	$x_{60}, x_{61}, x_{62}, x_{63}$
11	$HH_{03}^{1x} HH_{13}^{1x}$	$HG_{03}^{1x} HG_{13}^{1x}$	-	-	$x_{70}, x_{71}, x_{72}, x_{73}$
12	-	-	$GH_{20}^{1x} GH_{30}^{1x}$	$GG_{20}^{1x} GG_{30}^{1x}$	$x_{44}, x_{45}, x_{46}, x_{47}$
13	$HH_{20}^{1x} HH_{30}^{1x}$	$HG_{20}^{1x} HG_{30}^{1x}$	-	-	$x_{54}, x_{55}, x_{56}, x_{57}$
14	-	-	$GH_{21}^{1x} GH_{31}^{1x}$	$GG_{21}^{1x} GG_{31}^{1x}$	$x_{64}, x_{65}, x_{66}, x_{67}$
15	$HH_{21}^{1x} HH_{31}^{1x}$	$HG_{21}^{1x} HG_{31}^{1x}$	-	-	$x_{74}, x_{75}, x_{76}, x_{77}$
16	-	-	$GH_{22}^{1x} GH_{32}^{1x}$	$GG_{22}^{1x} GG_{32}^{1x}$	$y_{00}, y_{01}, y_{02}, y_{03}$
17	$HH_{22}^{1x} HH_{32}^{1x}$	$HG_{22}^{1x} HG_{32}^{1x}$	-	-	$y_{10}, y_{11}, y_{12}, y_{13}$
18	-	-	$GH_{23}^{1x} GH_{33}^{1x}$	$GG_{23}^{1x} GG_{33}^{1x}$	$y_{20}, y_{21}, y_{22}, y_{23}$
19	$HH_{23}^{1x} HH_{33}^{1x}$	$HG_{23}^{1x} HG_{33}^{1x}$	-	-	$y_{30}, y_{31}, y_{32}, y_{33}$
20	-	-	$GH_{00}^{1y} GH_{10}^{1y}$	$GG_{00}^{1y} GG_{10}^{1y}$	$HH_{00}^{1x}, HH_{01}^{1x}, HH_{02}^{1x}, HH_{03}^{1x}$
21	$HH_{00}^{1y} HH_{10}^{1y}$	$HG_{00}^{1y} HG_{10}^{1y}$	-	-	$HH_{10}^{1x}, HH_{11}^{1x}, HH_{12}^{1x}, HH_{13}^{1x}$
22	-	-	$GH_{01}^{1y} GH_{11}^{1y}$	$GG_{01}^{1y} GG_{11}^{1y}$	$HH_{20}^{1x}, HH_{21}^{1x}, HH_{22}^{1x}, HH_{23}^{1x}$
23	$HH_{01}^{1y} HH_{11}^{1y}$	$HG_{01}^{1y} HG_{11}^{1y}$	-	-	$HH_{30}^{1x}, HH_{31}^{1x}, HH_{32}^{1x}, HH_{33}^{1x}$
24	-	-	$GH_{00}^{2x} GH_{10}^{2x}$	$GG_{00}^{2x} GG_{10}^{2x}$	$y_{04}, y_{05}, y_{06}, y_{07}$
25	$HH_{00}^{2x} HH_{10}^{2x}$	$HG_{00}^{2x} HG_{10}^{2x}$	-	-	$y_{14}, y_{15}, y_{16}, y_{17}$
26	-	-	$GH_{01}^{2x} GH_{11}^{2x}$	$GG_{01}^{2x} GG_{11}^{2x}$	$y_{24}, y_{25}, y_{26}, y_{27}$
27	$HH_{01}^{2x} HH_{11}^{2x}$	$HG_{01}^{2x} HG_{11}^{2x}$	-	-	$y_{34}, y_{35}, y_{36}, y_{37}$
⋮	⋮	⋮	⋮	⋮	⋮

## 7 Results and Discussion

### 7.1 HARDWARE SIMULATION

An VHDL code is developed for the hardware efficient parallel FIR filters and 2D DWT, and the corresponding simulation is carried out by MODELSIM. The synthesis is done using XILINX XC3S5000-4FG1156. The simulation results for FFA structures, ISCA structures, 2-stage parallelism structures followed 2D DWT are shown in fig 7.1-7.15

#### 7.1.1 Simulation results of FFA structures

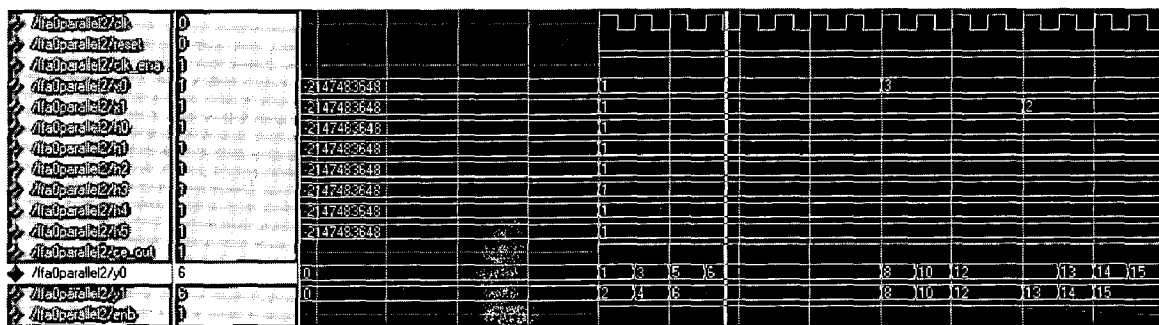


Fig 7.1 FFA0 2-parallel FIR filter

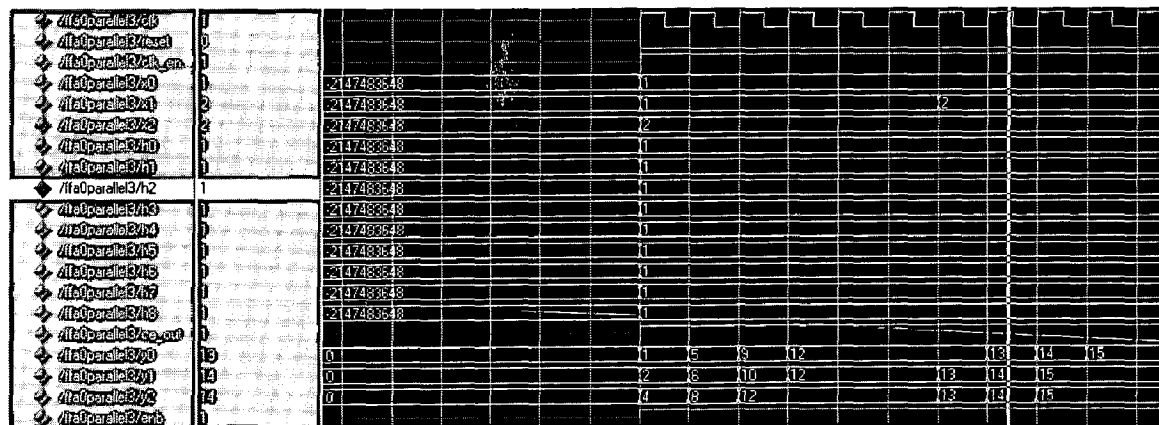


Fig 7.2 FFA0 3-parallel FIR filter



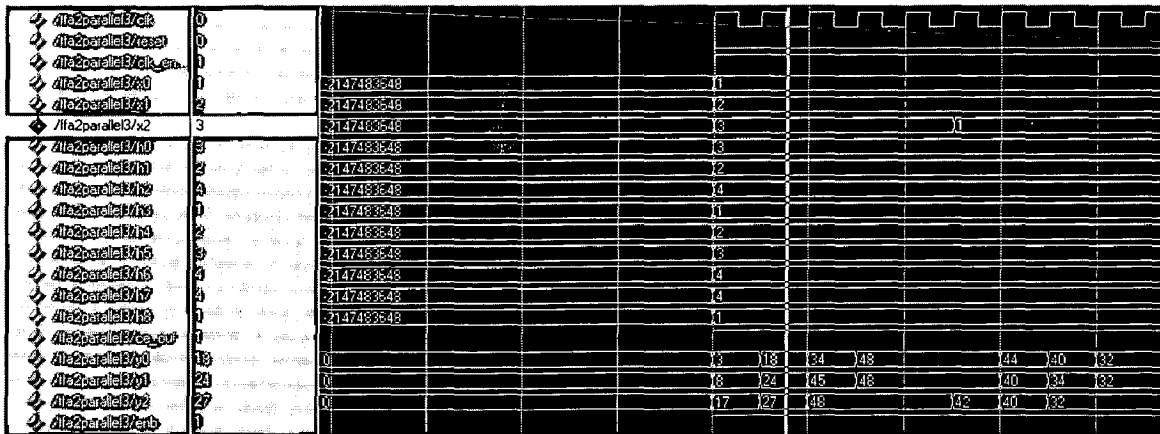


Fig 7.3 FFA2 3-parallel FIR filter

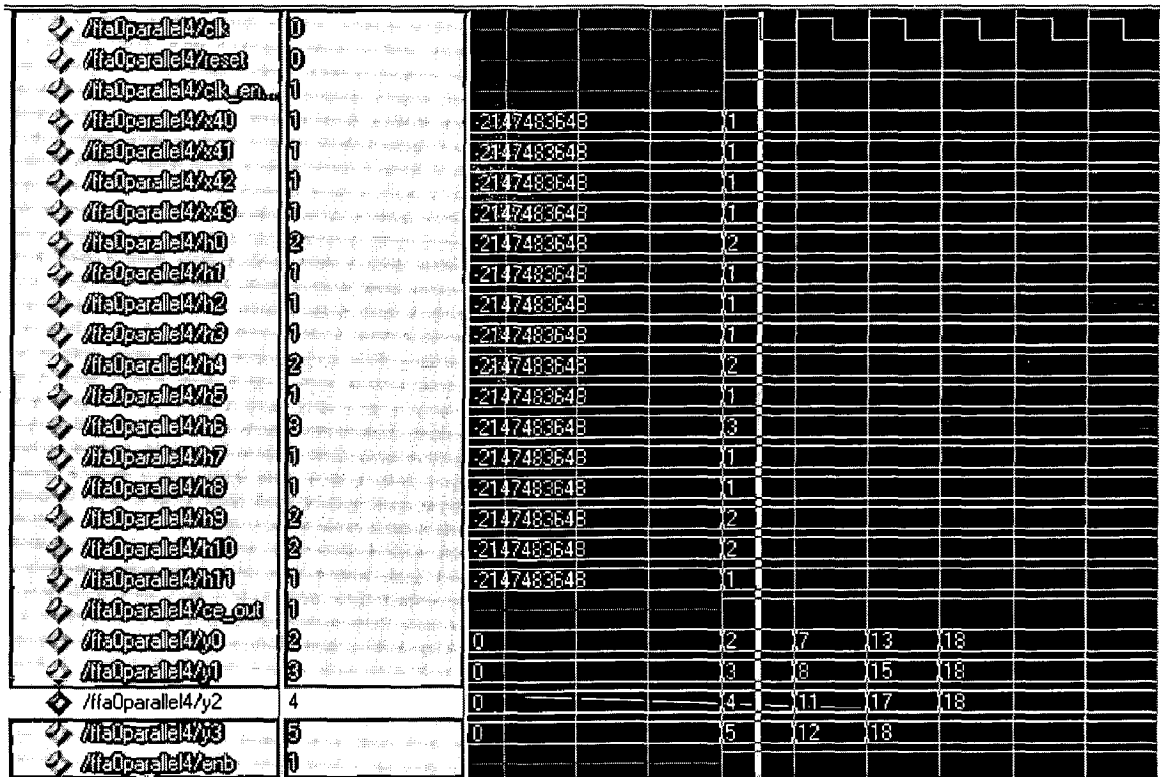


Fig 7.4 FFA0 4-parallel FIR filter

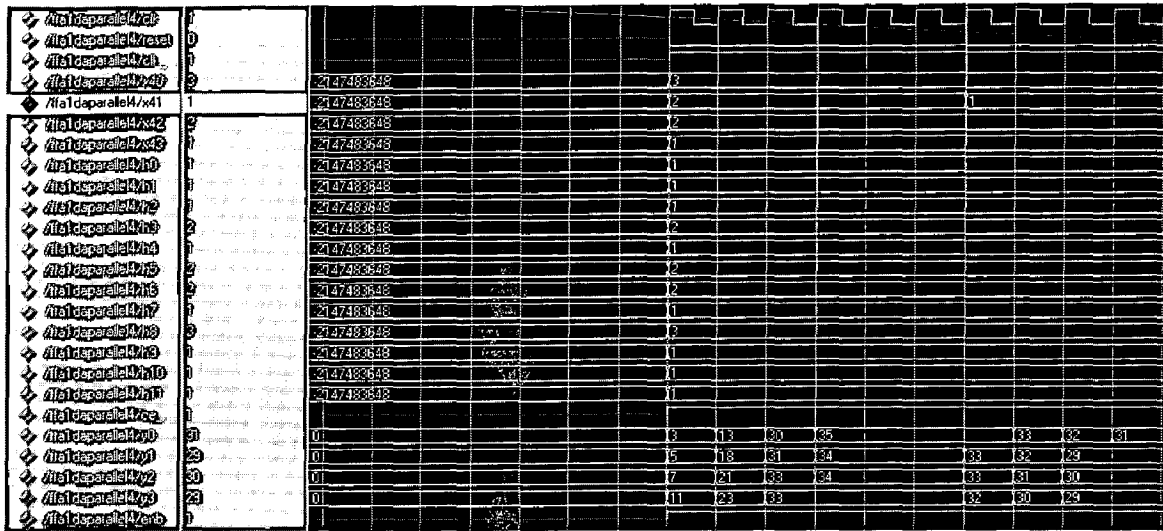


Fig 7.5 FFA1' 4-parallel FIR filter

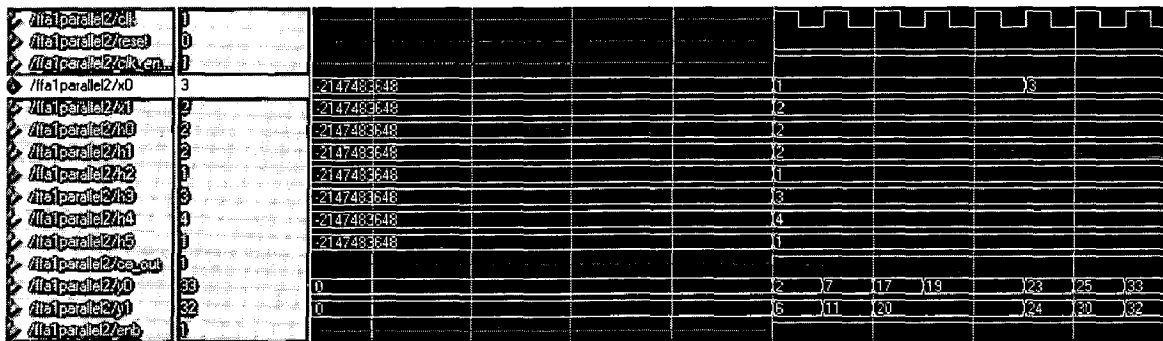


Fig 7.6 FFA1 2-parallel FIR filter

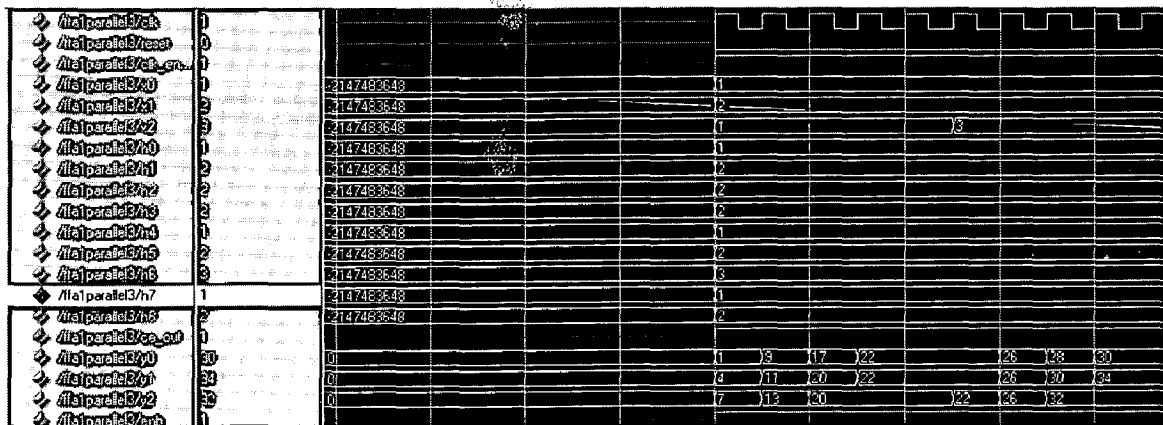


Fig 7.7 FFA1 3-parallel FIR filter

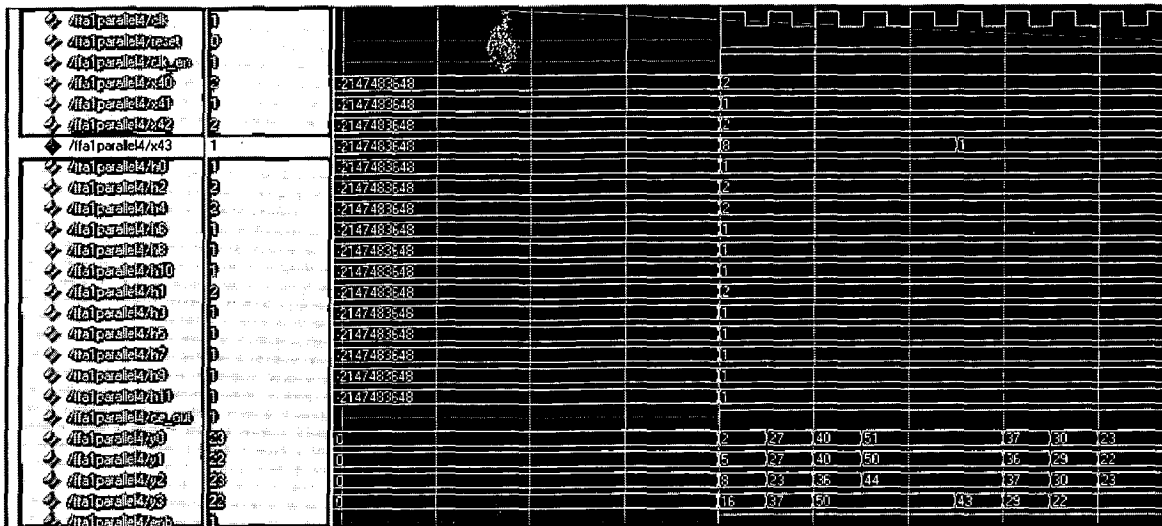


Fig 7.8 FFA1 4-parallel FIR filter

### 7.1.2 Simulation results of ISCA structures

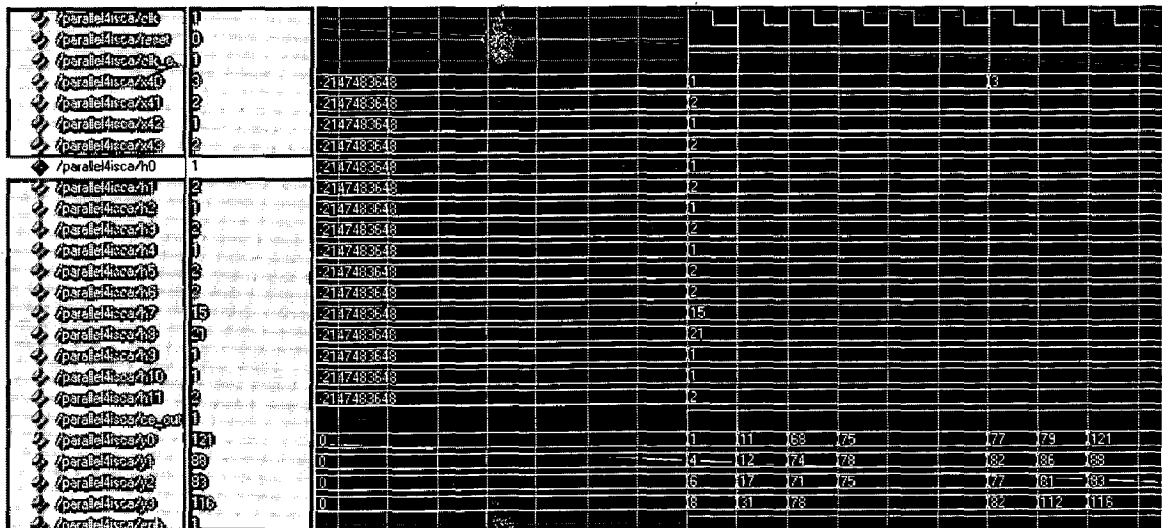


Fig 7.9 ISCA based 4-parallel FIR filter

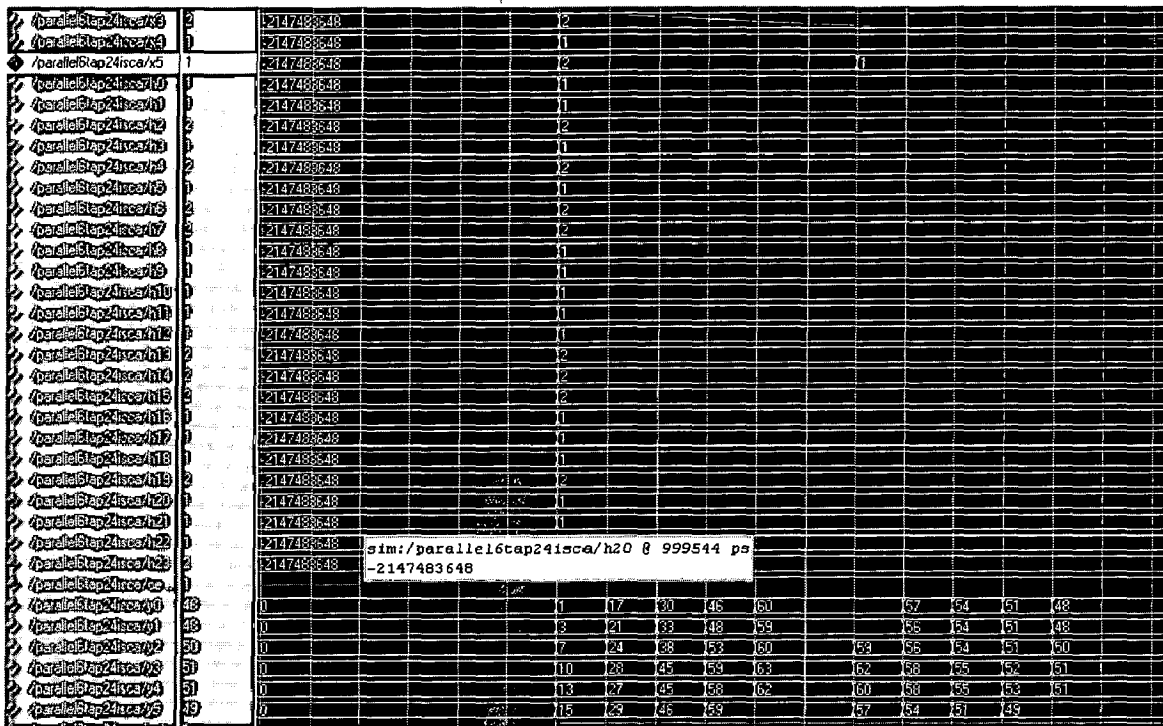


Fig 7.10 ISCA based 6-parallel 24 tap parallel FIR filter

### 7.1.3 Simulation results of 2-stage parallelism

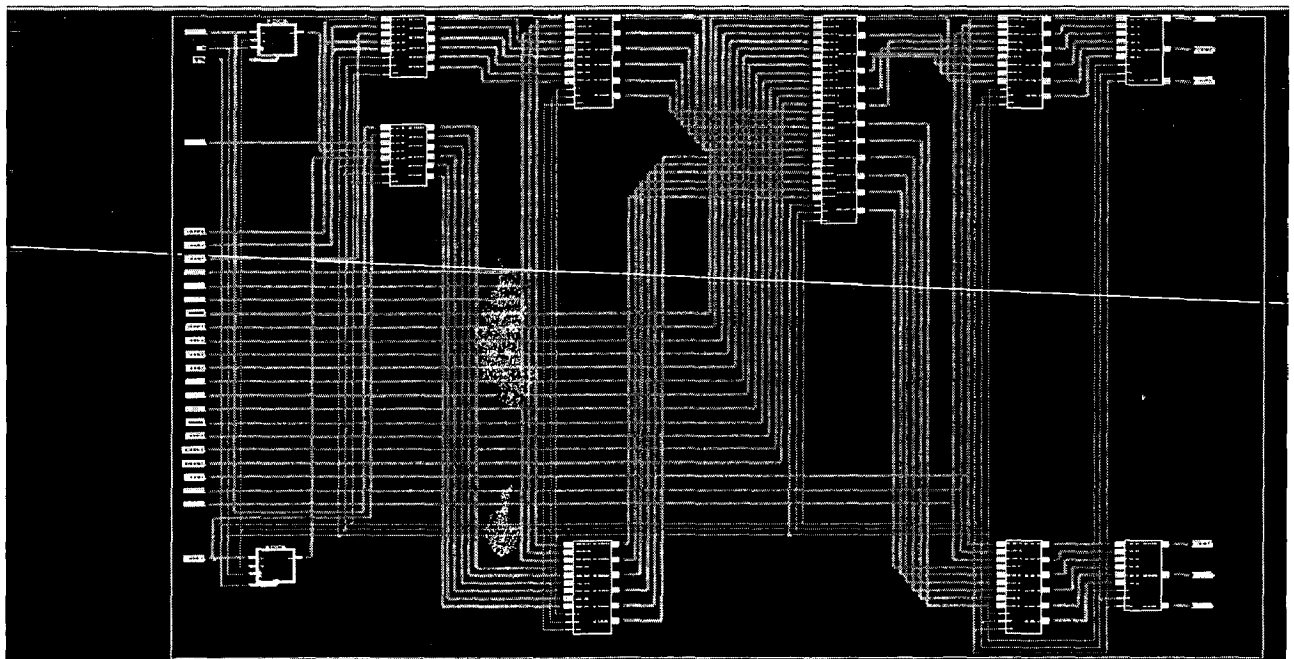


Fig 7.11.1 Schematic of 6-parallel 36-tap FIR filter



In the above figures 7.11-7.12, a latency of 12 clock cycles can be seen. This is due to the Delay Element Matrices for the transition of one stage to other stage of ISCA.

#### 7.1.4 Simulation results of 2D DWT based on parallel FIR filters

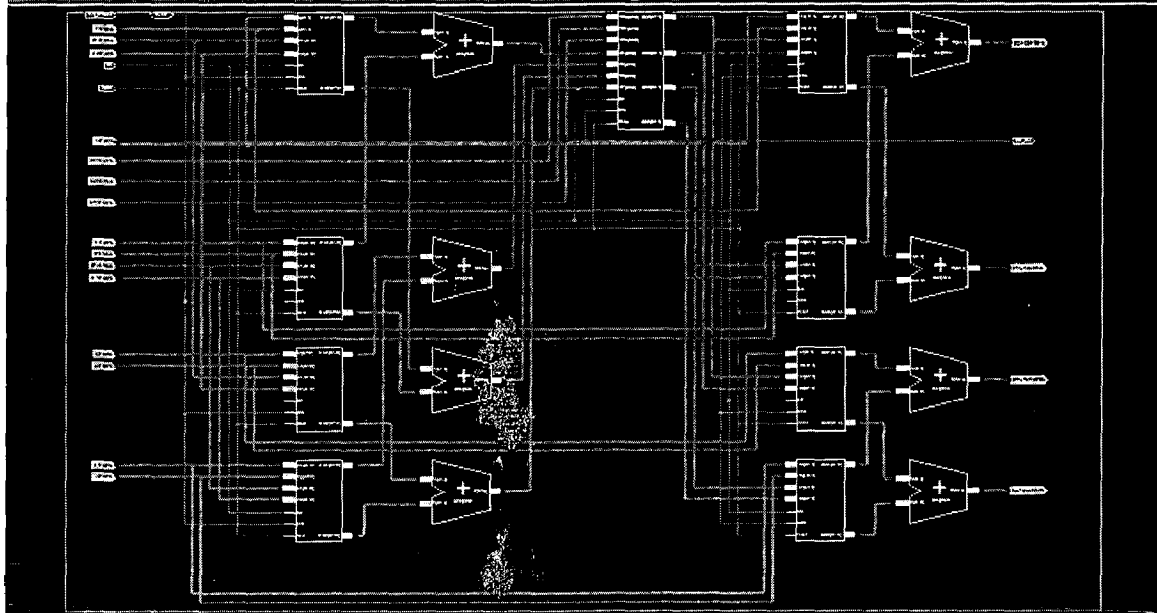


Fig 7.13 Schematic of 2D DWT for an  $8 \times 8$  image based on 2-parallel 2-tap FIR filter

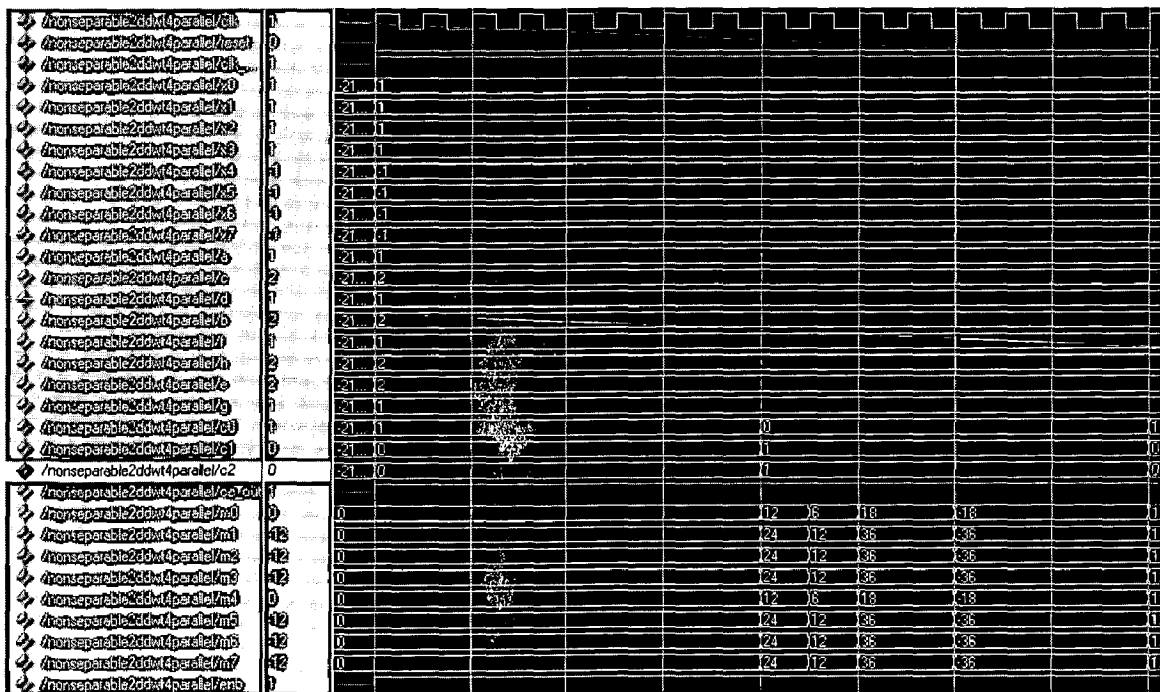
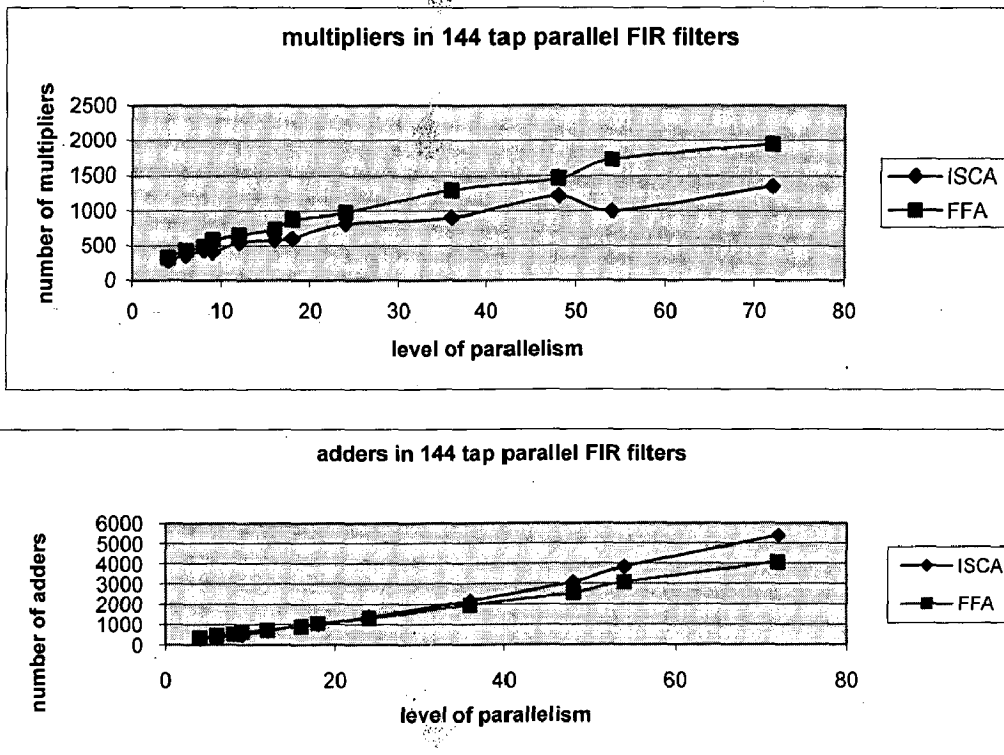


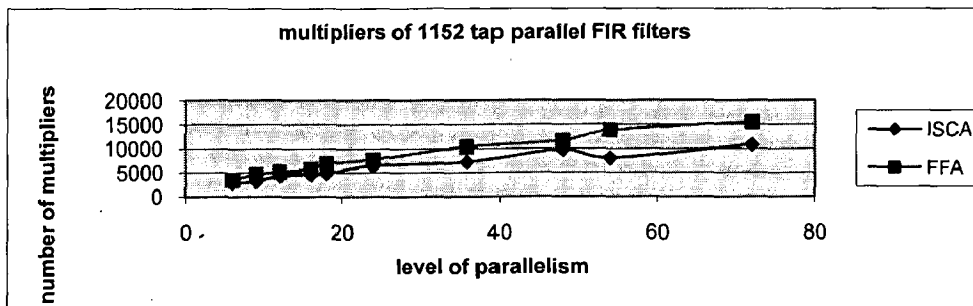
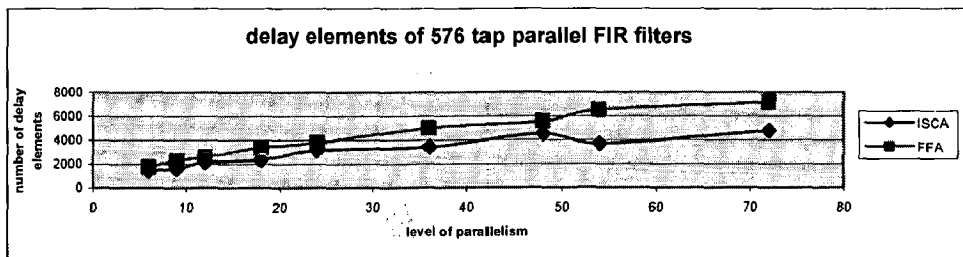
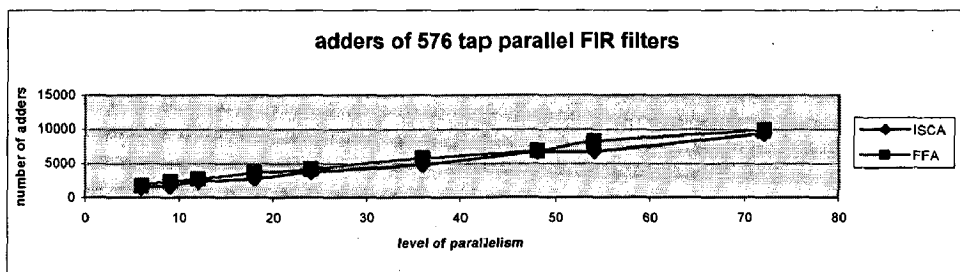
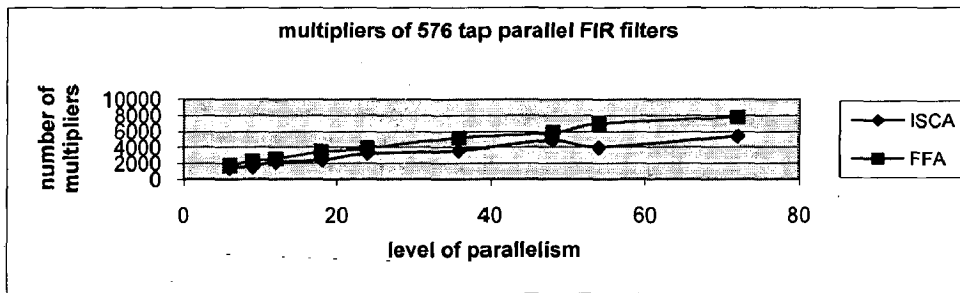
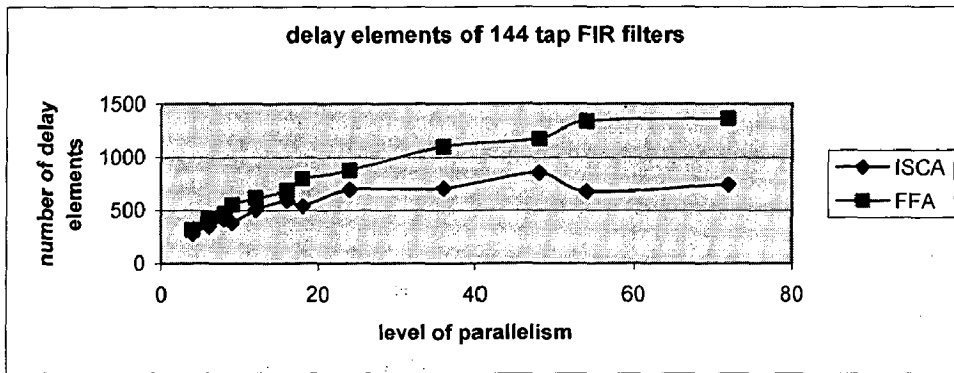
Fig 7.14 Non separable 2D DWT for an  $8 \times 8$  image based on 4-parallel 2-tap FIR filter

## 7.2 Comparison and Analysis

### 7.2.1 FFA VS ISCA

An Matlab code is developed for the comparison of this methods. Compared with FFA-based fast parallel FIR filter structures, ISC-based algorithm saves large amount of hardware cost. Although ISC-based algorithm uses more additions than FFA based structure with the increase of the level of parallelism, it can lead to large savings in the number of multiplications and delay elements. The additions will also decrease when the number of taps of FIR filter will be large. These are shown in figure 7.15 for different levels of parallelism. Note that the number of required additions is dependent on the order of iterations. The iteration order for short convolutions should be  $4 \times 4$ ,  $3 \times 3$  and  $2 \times 2$ , as this will lead to the lowest implementation cost; while, in FFA-based algorithm, the 2-parallel FFA is always applied first.







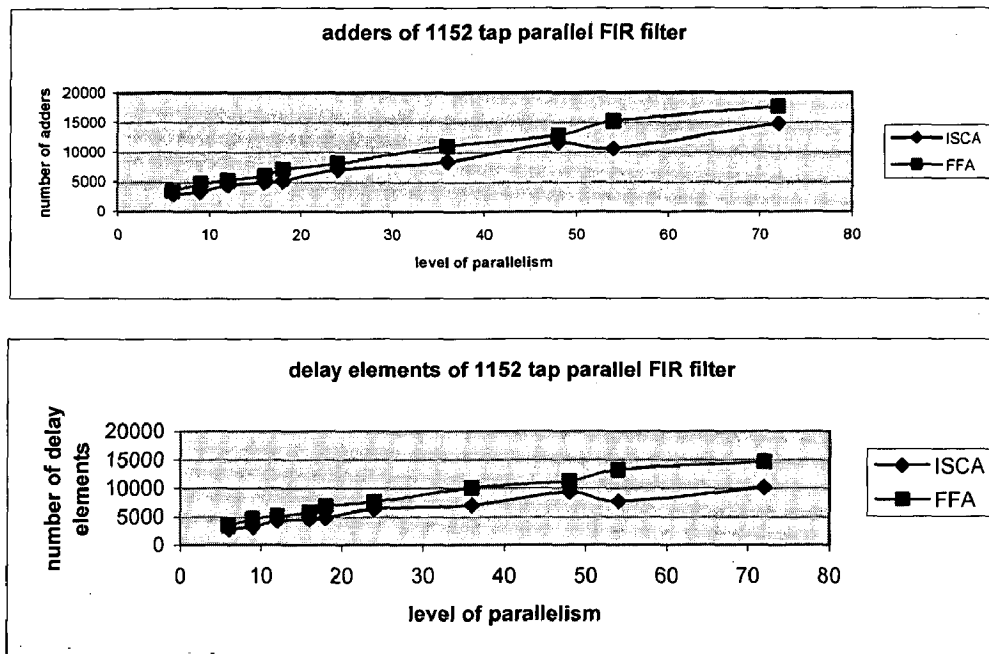


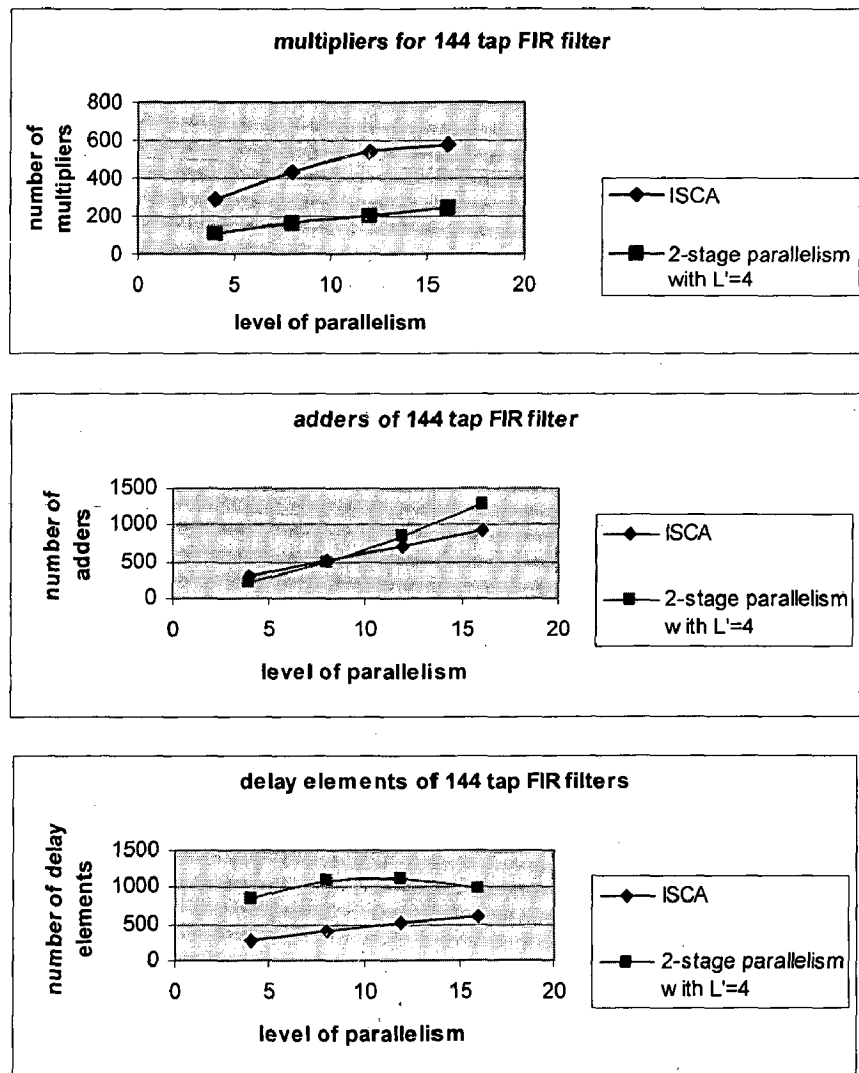
Fig 7.15 Hardware cost (i.e., complexity) comparison between ISCA and FFA for N=144, 576, 1152 at different levels of parallelism

### 7.2.2 ISCA Vs 2-stage Parallelism

Compared with ISCA-based fast parallel FIR filter structures, the improved structure can reduce large amount of hardware cost. 2-stage Parallelism can even reduce the hardware cost at the expense of more delay elements. Increasing the first stage parallelism will lead to more savings of required multiplications. Although direct implementation of the 2-stage parallel FIR structures (Method-1) will lead to large number of required delay elements, an interesting phenomenon of 2-stage parallel FIR structure (Method-1) is that the number of required multiplications is always less than or equal to the filter length  $N$  and doesn't increase as the parallelism level increases. A 2-stage  $L$ -parallel FIR structure has latency of  $2n$ , where  $n$  is defined in (5.1).

Latency is only decided by first stage parallelism and the 2-stage parallel FIR structure (Method-2) has efficiently controlled the increase of latency when  $L$  increases. Note that the ISCA-based parallel FIR design have no latency and  $L$  output data will be generated in the same clock cycle as the corresponding  $L$  input data are injected. In [9], linear

convolution-based processing core is used as a parallel FIR filter for the processing of subfilters. It requires the number of subfilters to be equal to the subfilter length. This requirement leads to irregular preprocessing and post-processing DEMs with low utilization efficiency and complex control signals. However, when the processing of subfilters is assigned to a parallel filter design, this restriction does not exist. Even if the filter length is not divisible by the parallelism level, zeros can be added at the end of the filter coefficients to make the total length divisible by the parallelism level. The computation results will be the same as the original filter and these are shown in figure 7.16



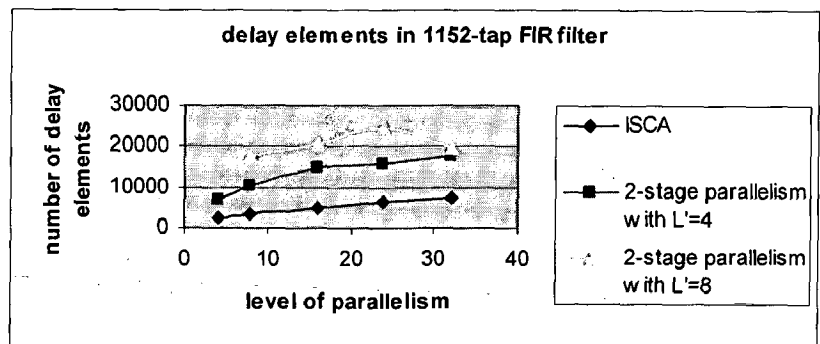
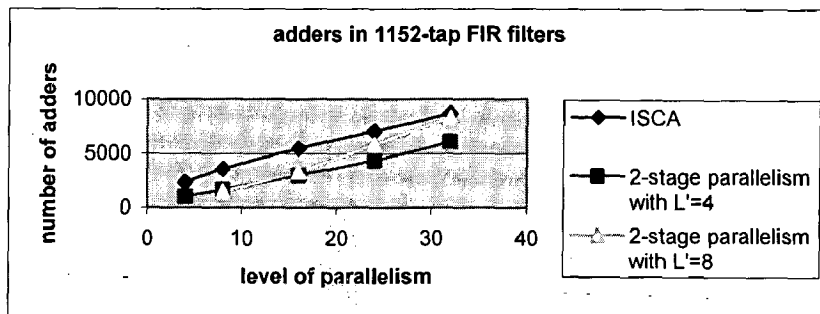
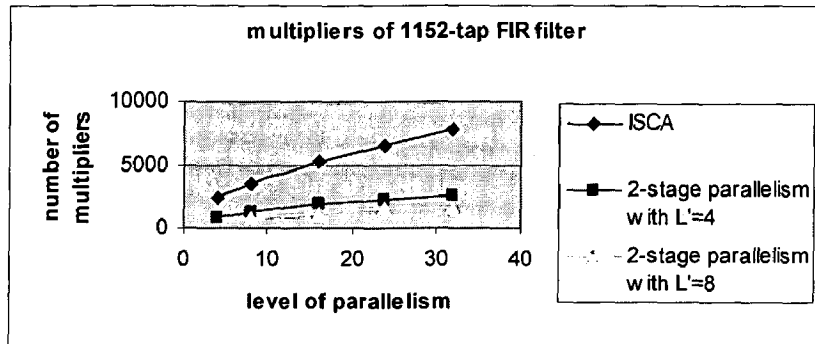


Fig 7.16 comparison of complexities of N=144,1152 between ISCA and 2-stage parallelism at different levels of parallelism.

### 7.2.3 Complexity for 2D DWT

This section compares the 2D DWT structure based on hardware efficient parallel FIR filter for the first resolution level with the recent convolution based architectures in [12],[13] and [15]. The following table compares the 2D DWT structure and previous convolution based architectures for an  $N \times N$  image with filter length 4, resolution level  $J=1$  and  $L$  gives the level of parallelism of 2-tap FIR filter

**Table 7.1**

**Comparison between 2D DWT structure and previous convolution based architectures for an  $N \times N$  image**

2-D DWT (convolution based architectures)	# of multipliers	# of adders	# of delay elements	Total Computing time in # clks
$N^2/8$ 2D DWT L=4	48	92	4N	$N^2/6$
$N^2/4$ 2D DWT L=2	24	32	2N	$N^2/3$
[12]	32	24	$N^2/4$	$N^2/3$
[13]	16	16	$N^2/4$	$2N^2/3$
[15]	30	30	0	$2N^2/3$

## CONCLUSION

These new algorithms like ISCA and 2-stage parallelism are very efficient in reducing hardware cost, especially when the length of the FIR filter is large. Tensor products are used to improve iterated short convolution algorithm in matrix form. Since preprocessing and postprocessing matrices are tensor products without delay elements, the ISCA facilitate automatic hardware implementation of parallel FIR filters, which is very efficient when the filter coefficients, word length or level of parallelism change, especially when the length of the FIR filter and the level of parallelism are large. The 2-stage parallel FIR structures also have regular structures and simple control signals, which utilizes the shared subfilters, is very easy for hardware implementation.

Hardware efficient parallel FIR filter structures are developed to speed up the processing speed of 2-D DWT and to control the increase of hardware cost at the same time. This design can be easily extended to achieve higher processing speed than the given highest processing speed with computing time of  $N^2/12$  cycles. This design is suitable for high-speed VLSI implementation of 2-D DWT because of its regular structure, simple control and 100% hardware utilization for continuous images.

## References:

- [1] J. I. Acha, "Computational structures for fast implementation of L-path and L-block digital filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 805–812, June 1989.
- [2] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," *J. VLSI Signal Processing Syst.*, vol. 17, no. 1, pp. 75–92, 1997.
- X [3] I.-S. Lin and S. K. Mitra, "Overlapped block-digital-filtering," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 586–596, Aug. 1996.
- [4] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.
- [5] J. G. Chung and K. K. Parhi, "Frequency-spectrum-based low-area low-power parallel FIR filter design," *EURASIP J. Appl. Signal Processing*, vol. 2002, no. 9, pp. 444–453, 2002.
- [6] Z. -J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 1322–1332, June 1991.
- [7] J. Granata, M. Conner, and R. Tolimieri, "A tensor product factorization of the linear convolution matrix," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1364–1366, Nov. 1991.
- [8] I.-S. Lin and S. K. Mitra, "Overlapped block digital filtering," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 43, pp. 586–596, Aug. 1996.
- [9] C. Cheng and K. K. Parhi, "Further complexity reduction of parallel FIR filters," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2005)*, Kobe, Japan, May 2005.
- [10] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, MA: Addison-Wesley, 1985.
- [11] S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants," *Math. Syst. Theory*, vol. 10, pp. 169–180, 1977.
- [12] Q. Dai, X. Chen, and C. Lin, "A novel VLSI architecture for multidimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 8, pp. 1105–1110, Aug. 2004.
- [13] P. C. Wu and L. G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 536–545, Apr. 2001.

- [14] C. Chakrabarti, M. Vishwanath, and R. Owens, "Architectures for wavelet transforms: A survey," *J. VLSI Signal Process.*, vol. 14, no. 2, pp. 171–192, Nov. 1996.
- [15] F. Marino, "Two fast architectures for the direct 2-D discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 49, no. 6, pp. 1248–1259, Jun. 2001.
- [16] F. Marino, "Efficient High-Speed/Low-Power Pipelined Architecture for the Direct 2-D Discrete Wavelet Transform", *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, pp1476-1491, vol 47, no.12, December 2000.
- [17] Chao Cheng, K. K. Parhi, "Hardware Efficient Fast Parallel FIR Filter Structures Based on Iterated Short Convolution," *IEEE Transactions. on Circuits and System-I: Regular Papers*, vol. 51, No.8, August 2004.
- [18] C. Cheng and K.K.Parhi, "Low-Cost Parallel FIR filter structures with 2-Stage Parallelism", *IEEE transactions on circuits and systems-I: Regular papers*, pp280-290, Vol-54, No.2, February 2007.
- [19] Richard. A. Haddad and Thomas, W.Parsons, "Digital Signal Processing Theory , Applications &Hardware" , Computer Science Press, 1994.
- [20] C. Cheng and K. K. Parhi, "Hardware efficient fast DCT based on novel cyclic convolution structures," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4419–4434, Nov. 2006.
- [21] T.K.Truong, I.S.Reed, R.G.Lipes and C.Wu , "On the Application of a Fast Polynomial Transform and the Chinese Remainder Theorem to Compute a Two-Dimensional Convolution" *IEEE transactions on Acoustics, Speech and Signal Processing*, vol-1, pp91-97, Feb 1981.
- [22] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 5, pp. 305–316, May 1995.
- [23] Y.-N. Chang and Li Yan-Sheng, "Design of highly efficient VLSI architectures for 2-D DWT and 2-D IDWT," in *Proc. IEEE Workshop on Signal Processing Systems*, Sep. 2001, pp. 133–140.
- [24] Proakis, "Digital Signal Processing", 3<sup>rd</sup> edition, John Wiley publications, 1999.

- [25] M. Winzker, "Low-power arithmetic for the processing of video signals," *IEEE Trans. on VLSI Systems*, vol. 6, no. 3, pp. 493–497, 1998.
- [26] J.-G. Chung, Y.-B. Kim, H.-J. Jeong, K. K. Parhi, and Z. Wang, "Efficient parallel FIR filter implementations using frequency spectrum characteristics," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 483–486, Monterey, Calif, USA, 31 May–3 June 1998.
- [27] C. Cheng and K.K.Parhi, "High speed VLSI implementation of 2D discrete wavelet transform", *IEEE transactions on Signal Processing*, pp393-493, Vol-56, No.1, Jan 2008.



## APPENDIX A : COOK-TOOM Algorithm

The Cook-Toom algorithm is a linear convolution algorithm for polynomial multiplication. The goal of this fast convolution algorithm is to reduce the multiplication complexity. One feature of the Cook-Toom method is it does not define a single solution; instead, it defines an entire family of methods for efficient multiplication. The method has considerable commonality with multiplication using the fast Fourier Transform, in that it works on the same principles of polynomial multiplication. The input numbers are split into limbs of a given size, and each is written as a polynomial, using the limb size as radix. Instead of multiplying the polynomials directly, they are evaluated at a set of points, and the values multiplied together at those points. The product polynomial is then determined, based on the products at those points. Finally, substitution of the radix returns the final answer. The degrees of freedom available to choose an appropriate algorithm are the number of limbs the input is divided into, and the points at which the polynomials are evaluated. It is based on Lagrange Interpolation Theorem, which states that

*Lagrange Interpolation Theorem:*

Let  $\beta_0, \dots, \beta_n$  be a set of  $n+1$  distinct points and let  $f(\beta_i)$ , for  $i = 0, 1, \dots, n$  be given.

There is exactly one polynomial  $f(p)$  of degree  $n$  or less that has value  $f(\beta_i)$  when evaluated at  $\beta_i$  for  $i = 0, 1, \dots, n$ . It is given by

$$f(p) = \sum_{i=0}^n f(\beta_i) \frac{\prod_{j \neq i} (p - \beta_j)}{\prod_{j \neq i} (\beta_i - \beta_j)} \quad (\text{A.1})$$

The linear convolution of an N-point sequence  $h = \{h_0, h_1, \dots, h_{N-1}\}$  and an L-point sequence  $x = \{x_0, x_1, \dots, x_{L-1}\}$ , which are expressed in polynomials as

$h(p) = h_{N-1}p^{N-1} + \dots + h_1p + h_0$  and  $x(p) = x_{N-1}p^{N-1} + \dots + x_1p + x_0$  respectively, is a polynomial of degree  $L+N-1$  expressed as  $s(p) = s_{L+N-2}p^{L+N-2} + \dots + s_1p + s_0$  can be computed by Cook-Toom algorithm as follows.

**Algorithm:**

1. Choose  $L+N-1$  different real numbers  $\beta_0, \beta_1, \dots, \beta_{L+N-2}$ .
2. Compute  $h(\beta_i)$  and  $x(\beta_i)$ , for  $i = 0, 1, \dots, L+N-2$ .
3. Compute  $s(\beta_i) = h(\beta_i)x(\beta_i)$ , for  $i = 0, 1, \dots, L+N-2$ .
4. Compute  $s(p)$  by Lagrange interpolation theorem, i.e., using equation (A.2), given by

$$s(p) = \sum_{i=0}^{L+N-2} s(\beta_i) \frac{\prod_{j \neq i} (p - \beta_j)}{\prod_{j \neq i} (\beta_i - \beta_j)} \quad (\text{A.2})$$

The Cook-Toom algorithm, in general, can also be expressed in matrix form as

$$s = Tx = CHDx$$

Where  $T$ =convolution matrix ,  $x, s$  are input and output matrices.

This algorithm provides a way to factorize the convolution matrix  $T$  into multiplication of one post addition matrix  $C$ , one diagonal matrix  $H$  with  $H_i, i = 0, 1, \dots, L+N-2$  on the main diagonal which determines the total number multiplications, and one preaddition matrix  $D$ . The Cook-Toom algorithm can reduce the complexity of multiplications from  $O(LN)$  to  $L+N-1$  at the expense of an increase in the number of additions. Further reduction of additions is obtained by the modification of the above algorithm.

The modified algorithm can be summarized as follows:

**Modified Algorithm:**

1. Choose  $L+N-2$  different real numbers,  $\beta_0, \beta_1, \dots, \beta_{L+N-3}$ .
2. Compute  $h(\beta_i)$  and  $x(\beta_i)$ , for  $i = 0, 1, \dots, L+N-3$ .
3. Compute  $s(\beta_i) = h(\beta_i)x(\beta_i)$ , for  $i = 0, 1, \dots, L+N-3$ .

4. Compute  $s'(\beta_i) = s(\beta_i) - s_{L+N-2}\beta_i^{L+N-2}$ , for  $i = 0, 1, \dots, L+N-3$ .
5. Compute  $s'(p)$  using Lagrange Interpolation Theorem, i.e., using equation (A.3), given by

$$s'(p) = \sum_{i=0}^{L+N-2} s'(\beta_i) \frac{\prod_{j \neq i} (p - \beta_j)}{\prod_{j \neq i} (\beta_i - \beta_j)} \quad (\text{A.3})$$

6. Compute  $s(p) = s'(p) + s_{L+N-2}p^{L+N-2}$ .

The Cook-Toom algorithm is efficient as measured by the number of multiplications. However, it is not efficient when the size of the problem increases, because for a large system, when the number of samples in the output sequence is large,  $\beta$  may take values other than  $0, \pm 1, \pm 2, \pm 4$ , etc. This may not result in simple preaddition and post addition matrices. For larger problems, the Winograd algorithm is more efficient.

## APPENDIX B : WINOGRAD Algorithm

The Winograd short convolution algorithm is based on the Chinese Remainder Theorem (CRT) over an integer/polynomical ring. The CRT over an integer ring is stated as

*Theorem B-1: CRT for Integers*

Given  $c_i = R_{m_i} [c]$ , for  $i = 0, 1, \dots, k$ , where  $m_i$  are moduli and are relatively prime, then

$$c = \sum_{i=0}^k c_i N_i M_i \text{ mod } M \quad (\text{B.1})$$

where  $M = \prod_{i=0}^k m_i$ ,  $M_i = M/m_i$  and  $N_i$  is the solution of

$$N_i M_i + n_i m_i = \text{gcd}(M_i, m_i) = 1 \quad (\text{B.2})$$

provided that  $0 \leq c \leq M$ . The notation  $R_{m_i} [c]$  represents the remainder when  $c$  is divided by  $m_i$ .

The CRT over an polynomial ring is stated as:

*Theorem B-2: CRT for Polynomials*

Given  $c^{(i)}(p) = R_{m^{(i)}(p)} [c(p)]$ , for  $i = 0, 1, \dots, k$ , where  $m^{(i)}(p)$  are relatively prime, then

$$c(p) = \sum_{i=0}^k c^{(i)}(p) N^{(i)}(p) M^{(i)}(p) \text{ mod } M(p) \quad (\text{B.3})$$

where  $M(p) = \prod_{i=0}^k m^{(i)}(p)$ ,  $M^{(i)}(p) = M(p)/m^{(i)}(p)$  and  $N^{(i)}(p)$  is the solution of

$$N^{(i)}(p) M^{(i)}(p) + n^{(i)}(p) m^{(i)}(p) = \text{gcd}(M^{(i)}(p), m^{(i)}(p)) = 1 \quad (\text{B.4})$$

provided that the degree of  $c(p)$  is less than the degree of  $M(p)$ .

To solve (B.2) and (B.4) for  $N_i$  and  $N^{(i)}(p)$ , one needs to use Euclidean GCD algorithm. Efficient convolution can be constructed using the CRT by choosing and factoring the polynomial  $M(p)$ . Based on the above theorems, the winograd convolution algorithm is summarized as follows:

**Algorithm :**

1. Choose the polynomial  $m(p)$  with degree higher than the degree of  $h(p)x(p)$  and factor into  $k+1$  relatively prime polynomials with real coefficients, i.e.,  $m(p) = m^{(0)}(p)m^{(1)}(p)\dots m^{(k)}(p)$ .
2. Let  $M^{(i)}(p) = M(p)/m^{(i)}(p)$  and use Euclidean GCD algorithm to solve  $N^{(i)}(p)M^{(i)}(p) + n^{(i)}(p)m^{(i)}(p) = \gcd(M^{(i)}(p), m^{(i)}(p)) = 1$  for  $N^{(i)}(p)$ .
3. Compute  $h^{(i)}(p) = h(p) \bmod m^{(i)}(p)$  and  $x^{(i)}(p) = x(p) \bmod m^{(i)}(p)$  for  $i = 0, 1, \dots, k$ .
4. Compute  $s^{(i)}(p) = h^{(i)}(p)x^{(i)}(p) \bmod m^{(i)}(p)$  for  $i = 0, 1, \dots, k$ .
5. Compute  $s(p)$ , using equation (B.5), given by

$$s(p) = \sum_{i=0}^k s^{(i)}(p) N^{(i)}(p) M^{(i)}(p) \bmod m(p) \quad (\text{B.5})$$

The number of multiplications is highly dependent on the degree of each  $m^{(i)}(p)$ . Therefore, the degree of  $m(p)$  should be as small as possible. According to CRT, the extreme case will be when  $\deg m(p) = (\deg s(p) + 1)$ . However, a more efficient form of Winograd algorithm can be obtained by choosing  $m(p)$  with a degree equal to that of  $s(p)$  and applying the CRT to  $s'(p) = s(p) - h_{L-1}x_{L-1}m(p)$ . Notice that  $s(p) \bmod m(p) = s'(p) \bmod m(p)$ .

The modified Winograd convolution algorithm is explained as follows

**Modified Algorithm:**

1. Choose a polynomial  $m(p)$  with degree equal to that of  $s(p)$  and factor into  $k+1$  relatively prime polynomials i.e.,  $m(p) = m^{(0)}(p)m^{(1)}(p)\dots m^{(k)}(p)$ .
2. Let  $M^{(i)}(p) = M(p)/m^{(i)}(p)$  and use Euclidean GCD algorithm to solve  $N^{(i)}(p)M^{(i)}(p) + n^{(i)}(p)m^{(i)}(p) = \gcd(M^{(i)}(p), m^{(i)}(p)) = 1$  for  $N^{(i)}(p)$ .
3. Compute  $h^{(i)}(p) = h(p) \bmod m^{(i)}(p)$  and  $x^{(i)}(p) = x(p) \bmod m^{(i)}(p)$  for  $i = 0, 1, \dots, k$ .
4. Compute  $s^{(i)}(p) = h^{(i)}(p)x^{(i)}(p) \bmod m^{(i)}(p)$  for  $i = 0, 1, \dots, k$ .
5. Compute  $s'(p)$ , using equation (B.6), given by

$$s'(p) = \sum_{i=0}^k s^{(i)}(p) N^{(i)}(p) M^{(i)}(p) \bmod m(p) \quad (\text{B.6})$$

6. Compute  $s(p) = s'(p) + h_{L-1}x_{L-1}m(p)$ .

## APPENDIX C: Efficient linear convolution examples

In this section, an efficient short length (i.e.,  $3 \times 3$  and  $4 \times 4$  linear convolutions) are addressed. These are derived from the modified Winograd algorithm.

### C-1 Short length linear convolutions:

An efficient  $3 \times 3$  linear convolution, by Winograd algorithm, can be obtained by choosing the polynomial  $m(p)$  as  $p(p-1)(p-2)(p+1)$ . Now

Step 1:

$$m^{(0)}(p) = p$$

$$m^{(1)}(p) = (p-1)$$

$$m^{(2)}(p) = (p-2)$$

$$m^{(3)}(p) = (p+1)$$

and  $M^{(i)}(p) = M(p)/m^{(i)}(p)$ , for  $i=0,1,2,3$  and using the relationship  $N^{(i)}(p)M^{(i)}(p) + n^{(i)}(p)m^{(i)}(p) = \gcd(M^{(i)}(p), m^{(i)}(p)) = 1$ , the following table is constructed.

Step 2:

$i$	$m^{(i)}(p)$	$M^{(i)}(p)$	$n^{(i)}(p)$	$N^{(i)}(p)$
0	$p$	$p^3 - 2p^2 - p + 2$	$1/2$	$-(p^2/2) + p + 1/2$
1	$p-1$	$p^3 - p^2 - 2p$	$-1/2$	$(1/2)(p^2 - 2)$
2	$p-2$	$p^3 - p$	$1/6$	$(-1/6)(p^2 + 2p + 3)$
3	$p+1$	$p^3 - 3p^2 + 2p$	$-1/6$	$(1/6)(p^2 - 4p + 6)$

Step 3:

$$\begin{aligned}
h^{(0)}(p) &= (h_2 p^2 + h_1 p + h_0) \bmod p = h_0 \\
h^{(1)}(p) &= (h_2 p^2 + h_1 p + h_0) \bmod (p-1) = h_0 + h_1 + h_2 \\
h^{(2)}(p) &= (h_2 p^2 + h_1 p + h_0) \bmod (p-2) = h_0 - h_1 + h_2 \\
h^{(3)}(p) &= (h_2 p^2 + h_1 p + h_0) \bmod (p+1) = h_0 + 2h_1 + 4h_2
\end{aligned}$$

and

$$\begin{aligned}
x^{(0)}(p) &= (x_2 p^2 + x_1 p + x_0) \bmod p = x_0 \\
x^{(1)}(p) &= (x_2 p^2 + x_1 p + x_0) \bmod (p-1) = x_0 + x_1 + x_2 \\
x^{(2)}(p) &= (x_2 p^2 + x_1 p + x_0) \bmod (p-2) = x_0 - x_1 + x_2 \\
x^{(3)}(p) &= (x_2 p^2 + x_1 p + x_0) \bmod (p+1) = x_0 + 2x_1 + 4x_2
\end{aligned}$$

Step 4:

$$\begin{aligned}
s^{(0)}(p) &= h_0 x_0 \\
s^{(1)}(p) &= (h_0 + h_1 + h_2)(x_0 + x_1 + x_2) \\
s^{(2)}(p) &= (h_0 - h_1 + h_2)(x_0 - x_1 + x_2) \\
s^{(3)}(p) &= (h_0 + 2h_1 + 4h_2)(x_0 + 2x_1 + 4x_2)
\end{aligned}$$

Step 5:

$$\begin{aligned}
s'(p) &= [s^{(0)}(p)N^{(0)}(p)M^{(0)}(p) + s^{(1)}(p)N^{(1)}(p)M^{(1)}(p) \\
&\quad + s^{(2)}(p)N^{(2)}(p)M^{(2)}(p) + s^{(3)}(p)N^{(3)}(p)M^{(3)}(p)] \\
&\quad \bmod(p^4 - 2p^3 - p^2 + 2p) \\
s'(p) &= p^3 \left[ (-1/2)s^{(0)}(p) - (1/2)s^{(1)}(p) + (1/6)s^{(2)}(p) - (1/6)s^{(3)}(p) \right] \\
&\quad + p^2 \left[ -s^{(0)}(p) + (1/2)s^{(1)}(p) + (1/2)s^{(3)}(p) \right] \\
&\quad + p^1 \left[ (1/2)s^{(0)}(p) + s^{(1)}(p) - (1/6)s^{(2)}(p) - (1/3)s^{(3)}(p) \right] \\
&\quad + p^0 \left[ s^{(0)}(p) \right] \bmod(p^4 - 2p^3 - p^2 + 2p) \\
s'(p) &= p^3 \left[ (-1/2)s^{(0)}(p) - (1/2)s^{(1)}(p) + (1/6)s^{(2)}(p) - (1/6)s^{(3)}(p) \right] \\
&\quad + p^2 \left[ -s^{(0)}(p) + (1/2)s^{(1)}(p) + (1/2)s^{(3)}(p) \right] \\
&\quad + p^1 \left[ (1/2)s^{(0)}(p) + s^{(1)}(p) - (1/6)s^{(2)}(p) - (1/3)s^{(3)}(p) \right] \\
&\quad + p^0 \left[ s^{(0)}(p) \right]
\end{aligned}$$



Step 6 :

$$\begin{aligned}
 s(p) &= s'(p) + h_2 x_2 (p^4 - 2p^3 - p^2 + 2p) \\
 s(p) &= p^3 \left[ (-1/2)s^{(0)}(p) - (1/2)s^{(1)}(p) + (1/6)s^{(2)}(p) - (1/6)s^{(3)}(p) \right] \\
 &\quad + p^2 \left[ -s^{(0)}(p) + (1/2)s^{(1)}(p) + (1/2)s^{(3)}(p) \right] \\
 &\quad + p^1 \left[ (1/2)s^{(0)}(p) + s^{(1)}(p) - (1/6)s^{(2)}(p) - (1/3)s^{(3)}(p) \right] \\
 &\quad + p^0 \left[ s^{(0)}(p) \right] + h_2 x_2 (p^4 - 2p^3 - p^2 + 2p)
 \end{aligned}$$

Therefore the final result in matrix form can be obtained as

$$S_5 = Q_3 \cdot \text{diag}(H_3) \cdot P_3 \cdot X_3$$

Where  $H_3 = \text{diag}[1/2 \ 1/2 \ 1/6 \ 1/6 \ 1] \cdot P_3 \cdot [h_0 \ h_1 \ h_2]^T$

$$Q_3 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & -2 & 2 \\ -2 & 1 & 0 & 3 & -1 \\ 1 & -1 & 1 & -1 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } P_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly, an  $4 \times 4$  convolution can be obtained by choosing the polynomial  $m(p)$  as  $p^2(p-1)(p+1)(p-2)(p+2)$ . The efficient  $4 \times 4$  short convolution in matrix form is given by

$$S_7 = Q_4 \cdot \text{diag}(H_4) \cdot P_4 \cdot X_4$$

where

$$Q_4 = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 4 & -4 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 4 & 4 & -2 & -2 & 4 \\ 5 & -5 & 5 & 4 & -4 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 & -1 & 2 & 2 & -5 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \\ 1 & 2 & 4 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} H_4 = \text{diag}[1/4 \ 1/4 \ 1/4 \ 1/6 \ 1/6 \ 1/48 \ 1/48 \ 1] \cdot P_4 \cdot [h_0 \ h_1 \ h_2 \ h_3]^T$$

Therefore it requires 8 multiplications i.e., lesser than FFA implementation.