Hardware Efficient Fast Parallel FIR Filter Structures Based On Iterated Short Convolution

Chao Cheng [1] *Keshab K. Parhi* [2]

[1] VIA Technologies (CHINA) INC., LTD., 6F, 9 Shangdi East Road, Haidian, Beijing, China, 100085

[2] Dept. of Electrical & Computer Engr., University of Minnesota, Minneapolis, MN 55455, USA

chaocheng@viatech.com.cn, parhi@ece.umn.edu

Abstract — This paper presents an Iterated Short Convolution (ISC) algorithm, based on the mixed radix algorithm and fast convolution algorithm. This ISC based linear convolution structure is transposed to obtain a new hardware efficient fast parallel FIR filter structure, which saves a large amount of hardware cost, especially when the length of the FIR filter is large. For example, for a 576-tap filter, the proposed structure saves 16.7% to 42.1% of the multiplications, 16.7% to 43.6% of the delay elements and 2.9% to 27% of the additions, which prior fast parallel structures use, when the level of parallelism varies from 6 to 72. These proposed structures exhibit regular structure.

1. INTRODUCTION

Many efforts have been directed towards deriving fast parallel filter structures in the past decade [1]-[3][5]-[7]. These algorithms usually first derive smaller length fast parallel filters and then cascade or iterate them to design parallel FIR filters with long block sizes.

An approach to increase the throughput of the signal rate with reduced complexity hardware was presented in [1]. This approach starts with the short convolution algorithms, which are transposed to obtain computationally efficient parallel filter structures. In [2], parallel FIR filters are implemented by using polyphase decomposition and fast FIR algorithms (**FFA**). The **FFA**'s are iterated to get fast parallel fir algorithm FIR Algorithms for larger block sizes.

Although, in [1] and [2], the small-sized parallel filter structures are computationally efficient, the number of required delay elements increases with the increase of the level of filtering parallelism. However, the transpose of the linear convolution structure in [1] is an optimal parallel FIR filter structure in terms of the required delay elements. While the Toeplitz-matrix factorization procedure in [1] buries additional delay elements inside the diagonal subfilter matrix and the algorithm in [2] places additional delay elements in the postaddition matrix, parallel FIR filtering structure based on the transpose of the linear convolution structure requires no additional delays inside the convolution matrix. Meanwhile, the positions of the delay elements in this transposed linear convolution structure are nicely placed and thus this structure is more regular.

In [3] a set of fast block filtering algorithms are derived based on fast short-length linear convolution algorithms to realize the parallel processing of subfilters, However, when convolution length increases, the number of additions increases dramatically, which leads to too complex preaddition and postaddition matrices to be practical for hardware implementation.

Therefore, if we could use fast convolution algorithms to decompose the convolution matrix with simple preaddition and postaddition matrices, we can get computationally efficient parallel FIR filter with the reduced number of required delay elements.

Fortunately, we can use Granata's [4] mixed radix algorithm, which decomposes the convolution matrix with tensor product into two short convolutions. This algorithm is combined with fast two and three point convolution algorithms to get the general iterated short convolution algorithm (**ISCA**). Although fast convolution of any length can be derived from Cook-Toom algorithm or Winograd algorithm [5], their preaddition or postaddition matrices may contain elements not in the set {1, 0, -1}, which makes them not suitable for hardware implementation of iterated convolution algorithm.

This paper is organized as follows. Section 2 investigates the iterated short convolution algorithm in matrix form. In section 3, **ISC** based new fast parallel FIR filter structures are presented. Section 4 presents the complexity comparisons of **ISC** based filters and existing FFA filters. Efficient short convolutions for the proposed hardware efficient fast parallel filters are defined in section 5. Algorithm analysis is done in section 6.

2. Iterated Short Convolution Algorithm

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 [5]. In this section mixed radix algorithm [4] is used to derive the generalized iterated short convolution algorithm using Tensor Product operator in matrix form.

A $M \times M$ $(M = mn)$ convolution can be decomposed into a $m \times m$ convolution and a $n \times n$ convolution, whose short convolution algorithms can be constructed with fast convolution algorithms such as Cook-Toom algorithm or Winograd algorithm [5] 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 the postaddition matrices. P_m and P_n are preaddition matrices. H_m and H_n are diagonal matrices, which can be denoted as $H_m = diag[P_m \times [h_0, h_1, ..., h_{m-1}]^T]$ and $H_n = diag[P_n \times [h_0, h_1, ..., h_{n-1}]^T]$ respectively. They determine the number of required multiplications in the iterated short convolution algorithm.

Using mixed radix algorithm [4], the resulting iterated short convolution algorithm can be represented as:

$$S_{2M-1} = A_{M\_mn}(Q_m \otimes Q_n) H_{M\_mn}(P_m \otimes P_n) X_M \qquad (2.1)$$

H_{M_mn} and A_{M_mn} are defined in (2.2) and (2.3), respectively.

$$H_{M\_mn} = diag[(P_m \otimes P_n)[h_0, h_1, ..., h_{M-1}]^T] \quad (2.2)$$

$$A_{M\_mn} = \begin{bmatrix} I_{(2n-1)\times(2n-1)} \\ I_{(2n-1)\times(2n-1)} \\ \ddots \\ I_{(2n-1)\times(2n-1)} \\ I_{(2n-1)\times(2n-1)} \end{bmatrix} \quad (2.3)$$

A_{M_mn} is a 2M-1 by (2m-1)(2n-1) matrix, composed of (2n-1) by (2n-1) unit matrices as defined in (2.3).

As an example, a 6×6 convolution can be decomposed into a 2×2 convolution and a 3×3 convolution. One set of fast convolution algorithms for 2×2 convolution and 3×3 convolution can be:

$S_3 = Q_2 H_2 P_2 X_2$ and $S_5 = Q_3 H_3 P_3 X_3$, respectively, where, $H_2 = diag(h_0, h_0 - h_1, h_1)$, $H_3 = diag(h_0, h_1, h_2, h_0 + h_1, h_0 + h_2, h_1 + h_2)$.

$$P_2 = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad P_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad Q_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

If we perform the 2×2 convolution first, the 6×6 convolution can be expressed in matrix form as in (2.4).

$$S_{11} = A_{6\_23}(Q_2 \otimes Q_3) H_{6\_23}(P_2 \otimes P_3) X_6 \quad (2.4)$$

where, $H_{6\_23} = diag[(P_2 \otimes P_3)[h_0, h_1, ..., h_5]^T]$, A_{6_23} is shown as

$$A_{6\_23} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.1 Representation for A_{6_23}

A $N\times N$ $(N = m\times n\times k)$ convolution can be further decomposed into three convolutions: $m\times m$, $n\times n$ and $k\times k$. One of the resulting iterated convolution algorithms can be represented by (2.5),

$$S_{2N-1} = A_{N\_Mk}(A_{M\_mn} \otimes I_{(2k-1)\times(2k-1)})(Q_m \otimes (Q_n \otimes Q_k)) H_{M\_mnk}(P_m \otimes (P_n \otimes P_k)) X_N \quad (2.5)$$

Where $H_{N\_mnk} = diag[(P_m \otimes (P_n \otimes P_k))[h_0, h_1, ..., h_{N-1}]^T]$

(2.5) is the proposed iterated short convolution algorithm that can be used to decompose a long convolution into any combination of short convolutions.

3. Fast Parallel FIR Algorithm Based On ISC

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

$$Y_L = P^T H_L Q^T A^T X_L \quad (3.1)$$

where, $Y_L = [Y_{L-1} \quad Y_{L-2} \quad \cdots \quad Y_0]^T$,

$X_L = [X_{L-1} \quad \cdots \quad X_1 \quad X_0 \quad z^{-L}X_{L-1} \quad \cdots \quad z^{-L}X_2 \quad z^{-L}X_1]^T$, X_i ,(i=0,1, …,L-1), represents the inputs x_{Lk+i},(k=0,1,2…);

$H_L = diag[P\times[H_0, H_1, \cdots, H_{L-1}]^T]$, H_i, (i =0,1,…,L-1) is the subfilter containing the coefficients h_{Lk+i}, (k=0,1,2…),

$$P = (P_{m_1\times n_1} \otimes (P_{m_2\times n_2} \otimes (\cdots (P_{m_{r-1}\times n_{r-1}} \otimes P_{m_r\times n_r}))))$$

$$P^T = (P^T_{m_1\times n_1} \otimes (P^T_{m_2\times n_2} \otimes (\cdots (P^T_{m_{r-1}\times n_{r-1}} \otimes P^T_{m_r\times n_r}))))$$

$$Q^T = (Q^T_{m_1\times n_1} \otimes (Q^T_{m_2\times n_2} \otimes (\cdots (Q^T_{m_{r-1}\times n_{r-1}} \otimes Q^T_{m_r\times n_r}))))$$

A^T is defined by (2.3) and (2.5), P^T and Q^T are preprocessing and postprocessing matrices, which determines the manner in which the inputs and outputs are combined, respectively [5].

Consider the 4-parallel FIR filter as an example. We start with the 4×4 convolution implemented with iterated two 2×2 short convolutions as:

$$Y_4 = (P_2^T \otimes P_2^T) H_4 (Q_2^T \otimes Q_2^T) A_{4\_22}{}^T X_4 \quad (3.2)$$

where, $H_4 = diag([H_0, H_0 - H_1, H_2, H_0 - H_2,$

$H_0 - H_1 - H_2 + H_3, H_1 - H_3, H_2, H_2 - H_3, H_3)]$,

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

This 4 parallel FIR architecture is shown in Figure 3.1.

4. Complexity Computation

The number of required multiplications is determined by the diagonal matrix H_L in (3.1) and given by (4.1)

$$M = \frac{N}{\prod_{i=1}^{r} L_i} \prod_{i}^{r} M_i \quad (4.1)$$

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} and N is the length of the filter.

Since each row of matrix A^T has only one "1", it won't increase the number of adders used. The number of required adders is determined by P^T and Q^T and is given by (4.2). Function $A(M_{m\times n})$ is the minimum number of adders used to calculate $M_{m\times n}X_n$, where $X_n = [x_1, x_2, \cdots x_n]^T$. Note that the number of required additions depends on the order of

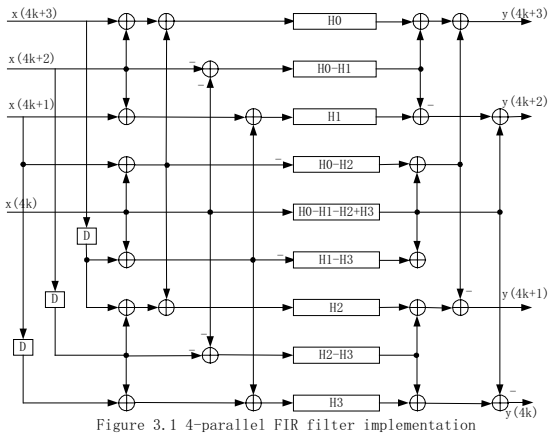

Figure 3.1 4-parallel FIR filter implementation

iteration; 3x3 convolution is iterated ahead of 2x2 convolution as this will lead to the lowest adder complexity.

The number of required delay elements is given by (4.3)

$$D = L - 1 + \prod_{i=1}^{r} M_i \left(\frac{N}{\prod_{i=1}^{r} L_i} - 1 \right) \tag{4.3}$$

In the following example, the number of multiplications and additions required to implement a 24-tap filter with 6-parallel filter is calculated with (4.1) and (4.2) respectively. The calculation is performed for $L_1 = 2, L_2 = 3$ form.

$$P_2^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}_{2\times 3}, \quad Q_2^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix}_{3\times 3} \tag{4.4}$$

$$P_3^T = \begin{bmatrix} 1&0&0&1&1&0 \\ 0&1&0&1&0&1 \\ 0&0&1&0&1&1 \end{bmatrix}_{3\times 6}, \quad Q_3^T = \begin{bmatrix} 1&-1&-1&0&0 \\ 0&-1&1&-1&0 \\ 0&0&-1&-1&1 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \end{bmatrix}_{6\times 5} \tag{4.5}$$

$$Y_{6P} = (P_2^T \otimes P_3^T) H_{2\times 3} (Q_2^T \otimes Q_3^T) A_{2\times 3}^T X_{6P} \tag{4.6}$$

$$M = (24/(2\times 3)) \times (3\times 6) = 72$$

$$A = (3\times 2 + 3\times 6) + (6\times 2 + 3\times 6) + (3\times 6)\left[24/(2\times 3) - 1\right] = 108$$

5. Efficient Short Convolution Definition

The computational efficiency of the proposed parallel FIR Filter Structures is depending on that of the selected short convolution algorithms. These efficient short convolution algorithms can lead to highly efficient parallel FIR Filter Structures and are defined in this section.

A. 3×3 short convolution

$$S_5 = Q_3 \cdot diag(H_3) \cdot P_3 \cdot X_3 \tag{5.1}$$

where,
$$Q_3 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & -2 & 2 \\ -2 & 1 & 0 & 3 & -1 \\ 1 & -1 & 1 & -1 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H_3 = diag\left[1/2 \;\; 1/2 \;\; 1/6 \;\; 1/6 \;\; 1\right] \cdot P_3 \cdot \left[h_0, h_1, h_2\right]^T$$

Therefore, we can get a new structure (5.2) for the computation of (4.5).

$$S_3 = P_3^T \cdot diag(H_3) \cdot Q_3^T \cdot X_3 \tag{5.2}$$

In (4.9), the elements for preprocessing and postprocessing matrices areextended and most of them are in the set {0, 2^k | k is integer}. Since multiplying a data by 2^k is to simply left shift this data by k bit, no hardware cost is increased, while maintaining the low complexity of the structure.

After sharing some additions, we can implement Q_3^T and P_3^T by 9 and 7 additions, respectively. Meanwhile, (5.2) requires 5 multiplications. Using this new structure, (4.6) can be computed by 60 multiplications and 106 additions. Therefore, for a 24-tap filter with 6-parallel filter, the new structure saves 12 multiplications, while using the same number of multiplication. (5.2) is more efficient than (4.5).

B. 4×4 short convolution.

$$S_5 = Q_4 \cdot diag(H_4) \cdot P_4 \cdot X_4 \tag{5.3}$$

where,
$$Q_4 = \begin{bmatrix} 4&0&0&0&0&0&0&0 \\ -4&4&-4&0&0&0&0&0 \\ -5&0&0&4&4&-2&-2&4 \\ 5&-5&5&4&-4&1&-1&0 \\ 1&0&0&-1&-1&2&2&-5 \\ -1&1&-1&-1&1&-1&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1&0&0&0 \\ 1&1&0&0 \\ 0&1&0&0 \\ 1&1&1&1 \\ 1&-1&1&-1 \\ 1&-2&4&-8 \\ 1&2&4&8 \\ 0&0&0&1 \end{bmatrix}$$

$$H_4 = diag\left[1/4\,1/4\,1/4\,1/6 \;\; 1/6 \;\; 1/48\,1/48\,1\right] \cdot P_4 \cdot \left[h_0, h_1, h_2, h_3\right]^T$$

The corresponding 4-parallel filter algorithm is (5.4)

$$Y_4 = P_4^T \cdot diag(H_4) \cdot Q_4^T \cdot X_4 \tag{5.4}$$

After sharing some additions, we can implement Q_4^T and P_4^T by 15 and 13 additions, respectively. Note that (5.4) requires 8 multiplications.

C. Another 4×4 short convolution.

Another 4×4 fast convolution algorithm can be constructed by inspection as: $S_7 = Q_4 \cdot diag(H_4) \cdot P_4 \cdot X_4$ (5.5)

where, $H_4 = diag\left(P_4 \cdot \left[h_0, h_1, h_2, h_3\right]^T\right)$

$$Q_4 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ -1&-1&0&0&1&0&0&0&0 \\ -1&1&-1&0&0&1&0&0&0 \\ 1&1&1&1&-1&-1&-1&-1&1 \\ 0&-1&1&-1&0&0&1&0&0 \\ 0&0&-1&-1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&0&0 \end{bmatrix}, P_4 = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&1 \\ 1&1&1&1 \end{bmatrix}$$

The corresponding 4-parallel filter algorithm is given by

$$Y_4 = P_4^T \cdot diag(H_4) \cdot Q_4^T \cdot X_4 \quad . \tag{5.6}$$

Q_4^T and P_4^T can implemented using 11 and 10 additions, respectively. When we use (5.5) as a short convolution for the proposed **ISC** based parallel filter, it is more efficient than iterating two 2×2 convolutions, Since (5.6) uses 1 less adder than (3.2). Therefore, when the parallel level has 4 as one of

$$A=\sum_{i=1}^{r}\left[(\prod_{j=1}^{i-1}n_j)(\prod_{k=i+1}^{r}m_k)A(P^T_{m_i\times n_i})\right]+\sum_{i=1}^{r}\left[(\prod_{j=1}^{i-1}n_j)(\prod_{k=i+1}^{r}m_k)A(Q^T_{m_i\times n_i})\right]+\prod_{i=1}^{r}M_i(\frac{N}{\prod_{i=1}^{r}L_i}-1) \qquad (4.2)$$

its factor, we use the structure (5.6).

6. Algorithm Analysis

Compared with **FFA** based fast parallel FIR filter structure, **ISC** based algorithm saves large amount of hardware cost. The number of required multiplications, additions and delay elements for a 576-tap filter are summarized for different levels of parallelism in Table 6.1.

The advantages of the proposed structures are obvious. From Table 6.1, we can see, the proposed **ISC** based algorithm can save 16.7% to 42.1% of the multiplications, 16.7% to 43.6% of the delay elements and 2.9% to 27% of the additions, which **FFA** based structure uses.

Note that the number of required additions is dependent on the order of iteration; the iterating order for short convolutions should be 4x4, 3x3 and 2x2, as this will lead to the lowest implementational cost; while, in FFA based algorithm, the 2-parallel FFA is always applied first [5].

7. Conclusion

In this paper, a novel iterated short convolution (**ISC**) based fast parallel FIR filter algorithm has been presented. This algorithm is 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. The proposed **ISC** based algorithm saves large amount of hardware cost. Since preprocessing and postprocessing matrices are tenor products without delay elements, this presented **ISC** based algorithm facilitates 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.

Reference

[1] *Acha, J.I., "*Computational structures for fast implementation of L -path and L -block digital filters," *Circuits and Systems, IEEE Transactions on*, Volume: 36 Issue: 6, June 1989 pp. 805 –812.

[2] Parker, D.A. and Parhi, K.K., "Area-efficient parallel FIR digital filter implementations," *Application Specific Systems, Architectures and Processors, 1996. ASAP 96, Proceedings of International Conference on*, 19-21 Aug. 1996 pp. 93 –111.

[3] Ing-Song Lin; Mitra, S.K.; *"*Overlapped block digital filtering," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, Volume: 43 Issue: 8 , pp. 586 –596, Aug. 1996.

[4] Granata, J.; Conner, M.; Tolimieri, R. *"*A tensor product factorization of the linear convolution matrix," *Circuits and Systems, IEEE Transactions on*, Volume: 38 Issue: 11, Nov. 1991 pp. 1364 –1366.

[5] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley and Sons, 1999.

[6] J.-G. Chung, Y.-B. Kim, H.-J. Jeong, and K. K. Parhi, "Efficient Parallel FIR Filter Implementations using Frequency Spectrum Characteristics", *Proc. of IEEE Int. Symp. on Circuits and Systems* , Monterey, May 31 - June 3, 1998.

[7] Z. -J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. on Signal Processing*, vol. 39, pp. 1322-1332, June 1991.

Table 6.1 the number of required multiplications (R.M.), Additions (R.A.) and delay elements (R.D.) for a 576-tap FIR filter.

L.	Algorithms		R.M. & saved M.		R.A. & saved A.		R.D. & saved D.	
6	ISC	3x3,2x2	1440	288	1486	266	1430	287
	FFA	2P,3P	1728	(16.7%)	1752	(15.2%)	1717	(16.7%)
9	ISC	3x3,3x3	1600	704	1721	637	1583	699
	FFA	3P,3P	2304	(30.6%)	2358	(27%)	2282	(30.6%)
12	ISC	4x4,3x3	2160	432	2326	362	2126	434
	FFA	2P,2P,3P	2592	(16.7%)	2688	(13.5%)	2560	(17%)
18	ISC	3x3,3x3,2x2	2400	1056	2807	847	2342	1049
	FFA	2P,3P,3P	3456	(30.6%)	3654	(23.2%)	3391	(30.9%)
24	ISC	4x4,3x3,2x2	3240	648	3805	419	3128	665
	FFA	2P,2P,2P,3P	3888	(16.7%)	4224	(9.92%)	3793	(17.5%)
36	ISC	4x4,3x3,3x3	3600	1584	4874	976	3410	1580
	FFA	2P,2P,3P,3P	5184	(30.6%)	5850	(16.7%)	4990	(31.7%)
48	ISC	4x4,4x4,3x3	4860	972	6733	203	4502	1046
	FFA	2P,2P,2P,2P,3P	5832	(16.7%)	6936	(2.9%)	5548	(18.9%)
54	ISC	3x3,3x3,3x3,2x2	4000	2912	6836	1426	3678	2845
	FFA	2P,3P,3P,3P	6912	(42.1%)	8262	(17.3%)	6523	(43.6%)
72	ISC	4x4,3x3,3x3,2x2	5400	2376	9428	490	4796	2399
	FFA	2P,2P,2P,3P,3P	7776	(30.6%)	9918	(5%)	7195	(33.3%)