Further Complexity Reduction of Parallel FIR Filters

Chao Cheng
Dept. of Electrical and Computer Engineering,
Univ. of Minnesota, Twin Cities
Minneapolis, MN 55455, USA
chao@ece.umn.edu

Keshab K. Parhi
Dept. of Electrical and Computer Engineering,
Univ. of Minnesota, Twin Cities
Minneapolis, MN 55455, USA
parhi@ece.umn.edu

***Abstract*— Based on recently published low complexity parallel FIR filter structures, this paper proposes a new scheme to further reduce their hardware complexity. FIR filter is firstly transformed into linear convolution, which is then implemented by Iterated Short Convolution algorithm. This linear convolution structure for FIR filter is used as a processing core to implement the subfitlers of previously proposed parallel FIR filter structures. Large amount of hardware can be saved by the new scheme. For example, for a 576-tap FIR filter, when the parallelism level changes from 12 to 72, the new scheme can save 1755 to 3375 multiplications at the cost of 21 to 4658 additions and 1516 to 4749 delay elements, respectively.**

I. INTRODUCTION

Fast parallel filter structures have been thoroughly discussed in [1-7]. 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.

Designs in [1-4] are based on polyphase decomposition, which buries additional delay elements inside the postaddition matrix and leads to large amount of delay elements and irregular structures when block sizes are long. However, the matrix form of linear convolution is used in [5-7], the delay element is regularly placed and the fast linear convolution algorithm can be used to reduce the hardware cost, especially the number of multiplications.

Recently, an Iterated Short Convolution (ISC) algorithm for long block size linear convolution was proposed and used to implement fast parallel FIR filter in [5]. This approach leads to large amount of hardware savings, compared to previous designs. According to the iterated short convolution algorithm in [5], a $L \times L$ ($L = L_1 L_2 \cdots L_r$) linear convolution can be first decomposed into r short convolutions, which can be computed by Cook-Toom or Winograd algorithm [3], $S_{2L_i-1} = Q_{L_i} H_{L_i} P_{L_i} X_{L_i}$ ($i = 1,2,\cdots r$), and then combined by (1.1) to get $L \times L$ linear convolution.

$$S_{2L-1} = A_L \cdot (Q_{L_1} \otimes (\cdots (Q_{L_{r-1}} \otimes Q_{L_r}))) \cdot H_L \cdot (P_{L_1} \otimes (\cdots (P_{L_{r-1}} \otimes P_{L_r}))) X_L \quad (1.1)$$

where, $H_L = diag[(P_{L_1} \otimes (\cdots (P_{L_{r-1}} \otimes P_{L_r})))[h_0, h_1, ..., h_{L-1}]^T]$

X_L is the input sequence $\{x_0, x_1, \cdots x_{L-1}\}$, and A_L is defined in [5].

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:

$$Y_L = P^T H_L Q^T A^T X_L \quad (1.2)$$

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…)

$H_L = diag[P \times [H_0, H_1, \cdots, H_{L-1}]^T]$,

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

H_i, (i =0,1,…,L-1) are the subfilters containing the coefficients h_{Lk+i},(k=0,1,2…),

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

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

When L is large, this iterated short convolution based parallel involves many subfilters, which contain many multiplication operations and the same hardware structure. If we can develop an efficient core to share the computation of all these subfilters in different time slots, we can save a lot of hardware cost.

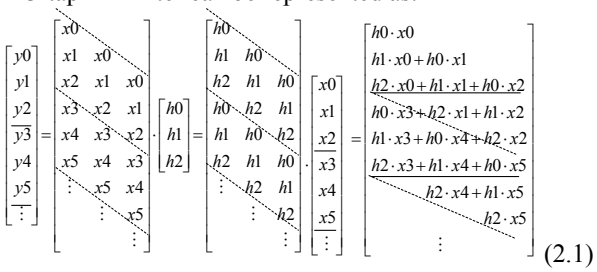
This paper is organized as follows. In section II, we will first give a scheme to transform N-tap FIR into NxN linear convolution, which generate N FIR filter outputs in just 1 clock cycle. In this process, the iterated short convolution algorithm in [5] will be used to reduce the hardware cost. Its application to reduce the hardware cost of previous FIR parallel filter design is discussed in the following sections. Section III presents the proposed new parallel FIR filter design. In section IV, we discuss the computational complexity of the new design. Obvious reduction of hardware cost will be shown, when a

comparison between the new design and previous one is carried in section V.

II. TRANSFORM FIR INTO LINEAR CONVOLUTION

We start with a simple example.

A 3-tap FIR filter can be represented as:



(2.1)

The computation of 3-tap FIR filter is transformed into that of 3x3 linear convolution algorithm. The last two rows of the five outputs of previous 3x3 linear convolution are summed with the upper two rows of the five outputs of the following convolution to get the 3 outputs of the 3-tap filter.

This property can be generalized, when an N-tap FIR filter is represented in the form of (2.1). The last N-1 rows of the 2N-1 outputs of previous NxN 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 N-tap FIR filter can be represented with NxN linear convolution and can be represented as:

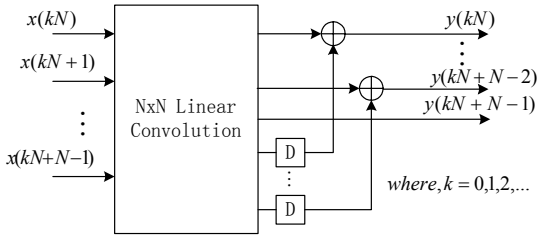


Fig. 2. 1. Implementation of N-tap FIR filter with NxN linear convoltution.

In Fig. 2.1, the hardware cost will depend on the complexity of the NxN linear convolution. Additional N-1 additions and N-1 delay elements are also required. We will use iterated short convolution algorithm in [5] to compute NxN linear convolution.

III. IMPROVED FAST PARALLEL FIR FILTER STRUCTURE

According to (1.2), 2-parallel 6-tap FIR filter with coefficients {h0, h1, h2, h3, h4, h5, h6} can be represented as:

$$Y_2 = P_2^T \cdot H_2 \cdot Q_2^T \cdot X_L \qquad (3.1)$$

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….

$H_2 = diag[\mathrm{H0} \quad \mathrm{H0-H1} \quad \mathrm{H1}]$; H0={h0, h2, h4},

H0-H1={h1-h1, h3-h2, h5-h3}, H1={h1, h3, h5};

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

It can be implemented as shown in Fig. 3.1.

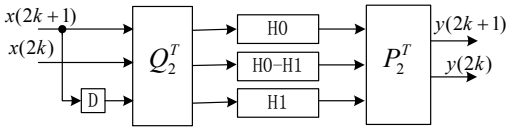


Fig. 3.1 implementation of 2-parallel FIR filter

In Fig. 3.1, H0, H0-H1, H1 are all 3-tap subfilters and require 9 multiplications, 6 additions and 6 delay elements. They have the same hardware structure, thus can be implemented, in consecutive time slots, with the same structure as shown in Fig. 2.1 (N=3).

We must first get the 3 consecutive inputs for each of these three 3-tap subfilters. These inputs can be first loaded from the first 6 sampling inputs, which will result in 3-clock delay before the filtering. When the outputs from the 3 subfilters H0, H0-H1 and H1 are computed in 3 consecutives with the structure in Fig. 2.1, 3 groups of new data are loaded. This process can be implemented with the following structure:

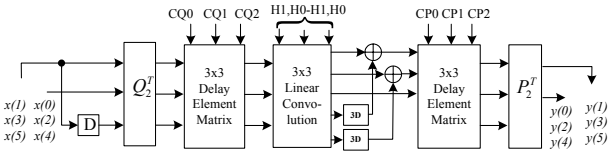


Fig. 3.2 (a) Proposed new 3-parallel FIR filter

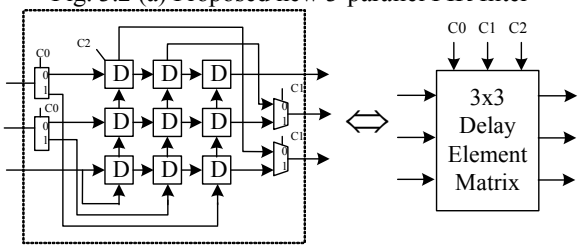


Fig. 3.2 (b) 3x3 Delay element Matrix

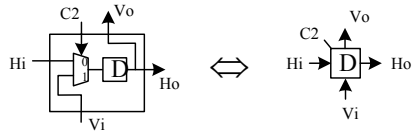


Fig. 3.2 (c) Delay element function

The data flow of the delay element matrix in Fig. 3.2 (b) 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 delay element matrix.

A 3x3 linear convolution can be implemented with fast linear convolution algorithm by 5 multiplications and 14 additions. The proposed structure for subfilters requires 5 multiplications, 16 additions and 24 delay elements. Thus

it saves 4 multiplications at the cost of 10 additions and 18 delay elements.

The proposed structure can compute 6 input data in 3 clock cycles, thus it has the same throughput rate as previous 2-parallel FIR filter structure.

The proposed structures for a given L-parallel N-tap FIR filter can be generalized as:

①Form an ISC based FIR filter by (1.2); ② Replace the subfilters with a core (N/L)x(N/L) linear convolution and two delay element matrices to arrange the input and output of the (N/L)x(N/L) linear convolution; ③ Implement the (N/L)x(N/L) linear convolution using the iterated short convolution algorithm.

IV. COMPLEXITY COMPUTATION

For a proposed L-parallel N-tap FIR filter, the computation is based on (N/L)x(N/L) linear convolution, which can be implemented with the iterated short convolution algorithm in [5].

The hardware complexity is computed as follows.

1) The number of required multiplications for the parallel FIR filter is equal to that of (N/L)x(N/L) linear convolution. If N/L can be decomposed as $L_1 L_2 \cdots L_s$, then the number of required multiplications can be given as:

$$M = \prod_{i=1}^{s} M_i \tag{4.1}$$

where, s is the number of $L_i \times L_i$ convolutions used to implement (N/L)x(N/L) linear convolution in (1.1), M_i is the number of multiplications used in the $L_i \times L_i$ convolution, which is determined by H_{L_i}.

2) The number of required additions is made up of three parts: ① additions used for the preprocessing and postprocessing matrices P^T and Q^T in (1.2); ② additions used to implement the (N/L)x(N/L) linear convolution by iterated short convolution (1.1); ③ additions used in Fig. 2.2 for transforming (N/L)-tap FIR filter into (N/L)x(N/L) linear convolution, which can be given by (N/L)-1. Therefore, the total number of required addition can be given by:

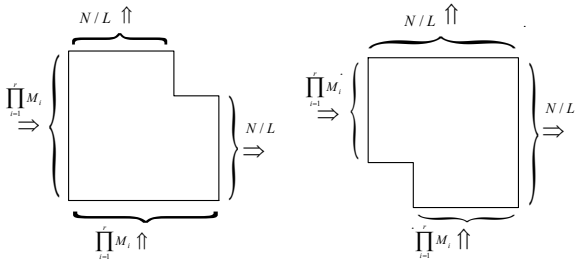
$$A = \sum_{i=1}^{r}\left[(\prod_{j=1}^{i-1} n_j)(\prod_{k=i+1}^{r} m_k) A(P^T_{m_i \times n_i})\right] + \sum_{i=1}^{r}\left[(\prod_{j=1}^{i-1} n_j)(\prod_{k=i+1}^{r} m_k) A(Q^T_{m_i \times n_i})\right]$$
$$+ \sum_{i=1}^{s}\left[(\prod_{j=1}^{i-1} n_j)(\prod_{k=i+1}^{s} m_k) A(P_{m_i \times n_i})\right]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

where, r is the number of $L_i \times L_i$ convolutions used to implement the L-parallel FIR filter in (1.2). Function $A(M_{m \times n})$ is the minimum number of adders used to calculate $M_{m \times n} X_n$, where $X_n = [x_1, x_2, \cdots x_n]^T$.

3) 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 (1.2); ② delay elements used in Fig. 2.2 for transforming (N/L)-tap FIR filter into (N/L)x(N/L) linear convolution, which can be given by $\left(\prod_{i=1}^{r} M_i\right) \cdot \left(\frac{N}{L} - 1\right)$; ③ delay elements used in the two delay element matrices, which can be noted as $2D_e$, where D_e is given by

$$D_e = \left(\prod_{i=1}^{r} M_i\right) \cdot \frac{N}{L} + \min\left((\prod_{i=1}^{r} M_i), \frac{N}{L}\right) \cdot \left|(\prod_{i=1}^{r} M_i) - \frac{N}{L}\right| \tag{4.3}$$

where, $\prod_{i=1}^{r} M_i$ is actually the number of output of matrix Q^T in (1.2) and is also the number of input data that go into the delay element matrix before the (N/L)x(N/L) linear convolution; it may be greater or less than N/L, and may be equal to N/L as in Fig. 3.2; when it is less than N/L, the shape of the delay element matrix is like Fig. 4.1 (a); when it is greater than N/L, the shape of the delay element matrix is like Fig. 4.1 (b).



(a) (b)

Fig 4.1 Shape of delay element matrix;

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

$$D = \left(\prod_{i=1}^{r} M_i\right) \cdot \left(\frac{N}{L} - 1\right) + 2D_e + L - 1 \tag{4.4}$$

V. IMPLEMENTATION ANALYSIS AND COMPLEXITY COMPARISON

As shown in the example of section II, the number of clock cycles required for the computation of the new structure depends on the number of subfilters of previous parallel FIR filter design. When the number of subfilters is too large, the proposed structure will have lower throughput rate, and thus may not be comparable to previous designs under the same criterion. Therefore, we may need to change the level of parallelism of the new structure, in order to control the number of subfilters to make sure the new structure and the previous one have the same throughput rate.

Here is another example. For a 4 parallel 24-tap FIR filter, the previous structures will use 6 clock cycles to compute 24 sampling data. But this 4-parallel filter has 9 subfilters. If we use this structure directly, the proposed design will require 9 clock cycles to compute 24 samples. In order to compensate this, we must use a 3-parallel structure, because it will have 6 subfilters. Then the length of the

core linear convolution will be 24/3=8, which is also the length of the subfilters in the 3-parallel structure. In this case, the new structure requires 3x9=27 multiplications, when 8 is decomposed as 2x4. However, the previous structure needs 9x6=54 multiplications, which is two times that of the new structure.

Compared with previous ISC based parallel FIR filter design, the new design can save large amount of hardware cost. A comparison in terms of required multiplications, additions and delay elements are summarized for different throughput rates in Table 5.1 for a 144-tap FIR filter and Table 5.2 for a 576-tap FIR filter. 'T' means the transpose forms of used short linear convolutions.

TABLE 5.1 NUMBER OF REQUIRED MULTIPLICATIONS (R.M.), ADDITIONS (R.A.) AND DELAY ELEMENTS (R.D.) FOR A 144-TAP FIR FILTER, IMPLEMENTED WITH DIFFERENT THROUGHPUT

Thro.	Algorithms		R.M.& saved M.		R.A. & saved A.		R.D.& saved D.	
4/N	ISC	L=4 (4T)	288	216	308	-129	283	-689
	Prop. ①	L=9 (3T, 3T) Ls=16 (4x4)	72		437		972	
6/N	ISC	L=6 (3T, 2T)	360	270	406	-119	350	-1145
	Prop. ②	L=8 (4T, 2T) Ls=18 (2x3x3)	90		525		1495	
8/N	ISC	L=8 (4T,2T)	432	297	509	-185	415	-1084
	Prop. ③	L=6 (3T, 2T) Ls=24 (2x3x4)	135		694		1499	
16/N	ISC	L=16 (4T, 4T)	576	306	926	-521	591	-861
	Prop. ④	L=4 (4T) Ls=36 (3x3x4)	270		1384 +63		1452	
18/N	ISC	L=18 (3T, 3T, 2T)	600	330	1007	-496	542	-765
	Prop. ⑤	L=4 (4T) Ls=36 (3x3x4)	270		1391 +112		1307	
24/N	ISC	L=24 (4T, 3T, 2T)	810	405	1375	-849	698	-666
	Prop. ⑥	L=3 (3T) Ls=48 (3x4x4)	405		2140 +84		1364	
48/N	ISC	L=48 (4T,4T,3T)	1215	405	3088	-1368	857	-203
	Prop. ⑦	L=2 (2T) Ls=72 (2x3x3x4)	810		4364 +92		1060	

In Table 5.1, ① and ② can preload input data in less clock cycles than the core linear convolution can finish processing all the previously loaded data. Thus, in these cases, delay element matrices will hold data for the linear convolution. Especially in the case of ①, the input data can be loaded into the delay element matrix in only 16 clock cycles, but 36 clock cycles are required to implement the 36 16-tap subfilters with the 16x16 core linear convolution hardware. Too many input data are loaded before they can be processed, which will lead to unnecessary large delay element matrices. We can actually divide the 36 16-size input data for the 16-tap subfilters into two groups, each with 18 16-size input data, and load them in consecutive two 16 cycles. The second group of data can be loaded in time before the linear core finishes processing the first group of data, so the processing speed will be maintained. This will reduce the number of required delay elements that are counted by (4.4) by 50%.

However, ④, ⑤, ⑥ and ⑦ will use more clock cycles to preload input data than the core linear convolution can finish processing all the previously loaded data. In order to maintain the processing speed, more preprocessing and postprocessing matrices are used in parallel. This will slightly increase the number of required additions that are counted by the (4.2).

TABLE 5.2 NUMBER OF REQUIRED MULTIPLICATIONS (R.M.), ADDITIONS (R.A.) AND DELAY ELEMENTS (R.D.) FOR A 576-TAP FIR FILTER, IMPLEMENTED WITH DIFFERENT THROUGHPUT

Thro.	Algorithms		R.M.& saved M.		R.A.& saved A.		R.D.& saved D.	
12/N	ISC	L=12 (4T, 3T)	2160	1755	2326	-21	2126	-4749
	Prop.	L= 12 (4T, 3T) Ls=48 (3x4x4)	405		2347		6875	
24/N	ISC	L=24 (4T,3T,2T)	3240	2430	3805	-826	3128	-4344
	Prop.	L=8 (4T, 2T) Ls=72 (2x3x3x4)	810		4461 +170		7472	
72/N	ISC	L=72(4T,3T,3T,2T)	5400	3375	9428	-4658	4796	-1516
	Prop.	L=4 (4T) L= 144 (3x3x4x4)	2025		13780 +306		6312	

From Table 5.1 and Table 5.2, we can see that with the increase of throughput, we can save more and more multiplications at cost of less and less delay elements. This is because the higher the throughput, the less the number of intermediate results needed to be held by delay elements. Although the number of additions also increases, it's reasonable, compared with the number of multiplications saved. Thus, the new design can save a large amount of hardware cost, compared with the previous parallel FIR design. This is especially true when the length of the FIR filter is long, as shown in Table 5.2.

REFERENCES

[1] D.A. Parker and K.K. Parhi, "Low-Area/Power Parallel FIR Digital Filter Implementations", *Journal of VLSI Signal Processing Systems*, Vol. 17, No.1, pp. 75-92, Sept. 1997.

[2] J.G. Chung and K.K. Parhi, "Frequency-Spectrum Based Low-Area Low-Power Parallel FIR Filter Design", *EURASIP Journal on Applied Signal Processing*, Vol. 2002, No. 9, pp. 444-453, 2002.

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

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

[5] 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.

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

[7] I.-S. Lin and S.K. Mitra, "Overlapped block digital filtering," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, Vol. 43, pp. 586 –596, Aug. 1996.