IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676, p-ISSN: 2320–3331, Volume 8, Issue 2 (Nov.-Dec.), PP 62-68
www.iosrjournals.org

Area-Efficient VLSI Implementation for Parallel Linear-Phase FIR Digital Filters of Odd Length Based on Fast FIR Algorithm

[1]Thodeti Madhukar, [2]M. R. N. Tagore, [3]Dr. Giri Babu Kande
[1]*PG Student (M.Tech VLSI), Dept. Of ECE, Vasireddy Venkatadri Ins. Tech., Nambur, Guntur, AP, India*
[2]*Associate Professor, Dept. Of ECE, Vasireddy Venkatadri Ins. Tech., Nambur, Guntur, AP, India*
[3] *Professor & Head, Dept. Of ECE, Vasireddy Venkatadri Ins. Tech., Nambur, Guntur, AP, India*

Abstract: *In this Paper, we propose a low-power multiplier design methodology which has multiplier-and-accumulator for high speed and by adopting the new Shift and adds implementation approach. This multiplier is designed by equipping the Power Suppression Technique on DA (distributed arithmetic) algorithm. The Shift and adder will avoid the unwanted addition and thus minimize the switching power dissipation, the performance was improved. When the DA algorithm is directly applied in FPGA (field programmable gate array) to realize FIR (finite impulse response) filter, it is easy to achieve the best configuration in the coefficient of FIR filter, the storage resources and the computing speed. The parallel FIR structures can lead to significant hardware savings for symmetric convolution in odd length from the existing FFA parallel FIR filter, particularly when the length of the filter is large. According to this, the paper provides the detailed analysis and discussion in the algorithm, the memory size and the look up table speed. Also, the corresponding optimization and improvement measures are discussed. The design results of simulation and test show that this method greatly reduces the FPGA hardware resource and the high speed filtering is achieved. The design has a big breakthrough compared to the traditional FPGA realization.*

Keywords: *DA Algorithm, Fast FIR algorithms (FFAs), parallel FIR, symmetric convolution, very large scale integration (VLSI).*

I. Introduction

With the recent rapid advances in multimedia and communication systems, real-time signal processing like audio signal processing, video/image processing, or large-capacity data processing are increasingly being demanded. The multiplier and multiplier-and-accumulator are the essential elements of the digital signal processing such as filtering, convolution, transfo-rmations and Inner products. There are different entities that one would like to optimize when designing a VLSI circuit. These entities can often not be optimized simultaneously, only improve one entity at the expense of one or more others The design of an efficient integrated circuit in terms of power, area, and speed simultaneously, has become a very challenging problem. Power dissipation& speed are recognized as critical parameters in modern the objective of a good multiplier is to provide a physically compact, good speed and low power consuming chip.

In the majority of digital signal processing (DSP) applications the critical operations usually involve many multiplications and/or accumulations. For real-time signal processing, a high speed and high throughput Multiplier-Accumulator (MAC) is always a key to achieve a high performance digital signal processing system. In the last few years, the main consideration of MAC design is to enhance its speed. This is because; speed and throughput rate is always the concern of digital signal processing system. But for the epoch of personal communication, low power design also becomes another main design consideration. This is because; battery energy available for these portable products limits the power consumption of the system. Therefore, the main motivation of this work is to investigate various Pipelined multiplier/accumulator architectures and circuit design techniques which are suitable for implementing high throughput signal processing algorithms and at the same time achieve low power consumption

The paper presents the improvement and optimization of the DA algorithm aiming at the problems of the configuration in the coefficient of FIR filter, the storage resource and the calculating speed, which make the memory size smaller and the operation aped faster to improve the computational performance.

II. FIR Multiplier

FIR multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 FIR recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To FIR recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous

block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 1 shows the grouping of bits from the multiplier term for use in modified FIR encoding.

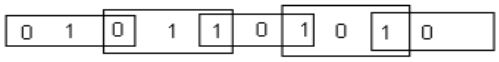

Figure 1 Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified FIR algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1.

Table 1 Operation on the Multiplicand

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

For the partial product generation, we adopt Radix-4 Modified FIR algorithm to reduce the number of partial products for roughly one half. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits.

Figure 2, shows a computing example of FIR multiplying two numbers"2AC9" and "006A". The shadow denotes that the numbers in this part of FIR multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals.

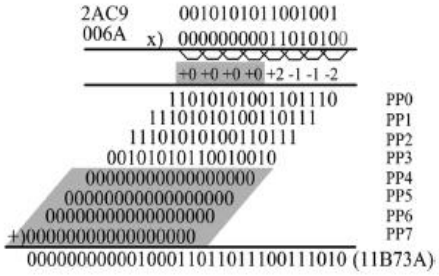

Figure 2 Illustration of multiplication using modified FIR encoding

The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the FIR encoding results of the operand B. When the operand besides the FIR encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

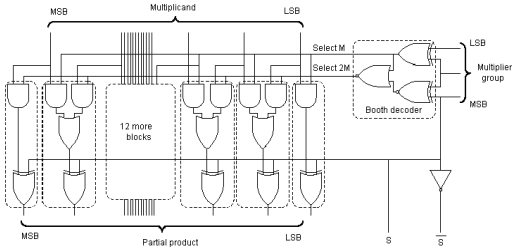

Figure 3 FIR partial product selector logic

The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2's complement it is negative.
The partial product is generated by doing AND between 'a' and 'b' which are a 4 bit vectors as shown in fig.3 If we take, four bit multiplier and 4-bit multiplicand we get sixteen partial products in which the first partial product is stored in 'q'. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

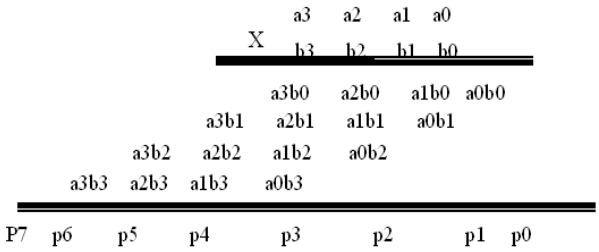

Figure 4 FIR partial products Generation

The multiplication second step reduces the partial products from the preceding step into two numbers while preserving the weighted sum. The sough after product P is the sum of those two numbers. The two numbers will be added during the third step The "Wallace trees" synthesis follows the Dadda's algorithm, which assures of the minimum counter number. If on top of that we impose to reduce as late as (or as soon as) possible then the solution is unique. The two binary number to be added during the third step may also be seen a one number in CSA notation (2 bits per digit).

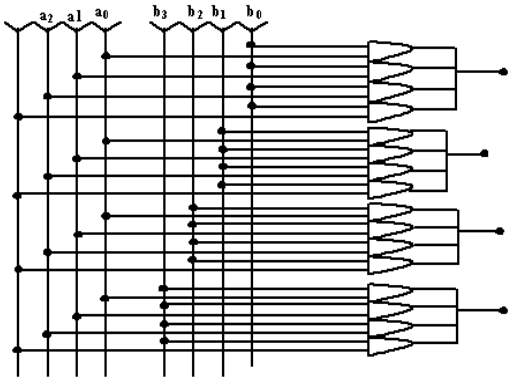

Figure 5 FIR single partial product selector logic

Multiplication consists of three steps: 1) the first step to generate the partial products; 2) the second step to add the generated partial products until the last two rows are remained; 3) the third step to compute the final multiplication results by adding the last two rows. The modified FIR algorithm reduces the number of partial products by half in the first step. We used the modified FIR encoding (MBE) scheme proposed in. It is known as the most efficient FIR encoding and decoding scheme. To multiply X by Y using the modified FIR algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table II shows the rules to generate the encoded signals by MBE scheme and Fig. 6 (a) shows the corresponding logic diagram. The FIR decoder generates the partial products using the encoded signals as shown in Fig. 6

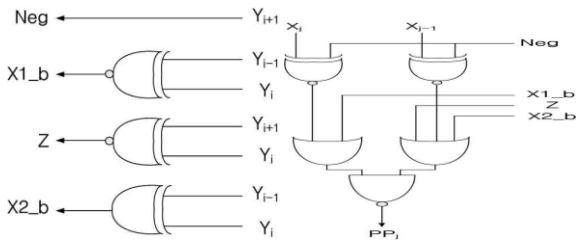

Figure 6.a FIR Encoder **6.b FIR Decoder**

The partial products generated by the modified FIR algorithm are added in parallel using the Wallace tree until the last two rows are remained. The final multiplication results are generated by adding the last two rows. The carry propagation adder is usually used in this step.

Y_{i+1}	Y_i	Y_{i-1}	Value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

Figure 7 Truth Table of MBE scheme

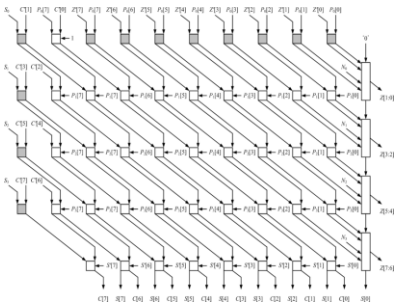

Figure 8 Architecture of the Proposed CSA tree

The architecture of the hybrid-type CSA that complies with the operation of the proposed MAC is shown in Fig. 5, which performs 8-bit operation. In Fig. 2.11 Si is to simplify the sign expansion and Ni is to compensate 1's complement number into 2's complement number. S[i] and C[i] correspond to the i^{th} bit of the feedback sum and carry. Z[i] is the i^{th} bit of the sum of the lower bits for each partial product that were added in advance and Z'[i] is the previous result. In addition, Pj[i] corresponds to the ith bit of the jth partial product. Since the multiplier is for 8 bits, totally four partial products are generated from the FIR encoder. This CSA requires at least four rows of FAs for the four partial products. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For an -bit MAC operation, the level of CSA is (n/2+1) . The white square in Fig. 2.11 represents an FA and the gray square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input

The critical path in this CSA is determined by the 2-bit CLA. It is also possible to use FAs to implement the CSA without CLA. However, if the lower bits of the previously generated partial product are not processed in advance by the CLAs, the number of bits for the final adder will increase. When the entire multiplier or MAC is considered, it degrades the performance. In Table I, the characteristics of the proposed CSA architecture have been summarized and briefly compared with other architectures. For the number system, the proposed CSA uses 1'scomplement, but ours uses a modified CSA array without sign extension. The biggest difference between ours and the others is the type of values that is fed back for accumulation. Ours has the smallest number of inputs to the final adder.

III. Improved Design of The DA Algorithm

The principle of DA algorithm is as follows, the output of linear time-invariant system is shown as Eq. 1.

$$Y = \sum_{m=1}^{M} A_m X_m \qquad ---------- \quad Eq.1$$

Where A_m is a fixed factor, X_m is the input data ($|X_m|$<1). X_m can be expressed as Eq. 2 using the binary complement.

$$X_m = -x_{m0} + \sum_{n=1}^{N-1} x_{mn}\, 2^{-n} \text{ ---------- Eq. 2}$$

Where x_{mn} is 0 or 1, x_{m0} is sign bit; $x_{m\,N-1}$ is the least significant bit. Then Y can be expressed as Eq .3.

$$Y = \sum_{m=1}^{M} A_m \left(\sum_{n=1}^{N-1} x_{mn}\, 2^{-n} - x_{m0} \right)$$
$$= \sum_{n=1}^{N-1} \sum_{m=1}^{M} A_m x_{mn}\, 2^{-n} + \sum_{m=1}^{M} A_m (-x_{m0}) \text{ ----------} \qquad \text{Eq.3}$$

In Eq. 3, as the value of x_{mn} is 0 or 1, there are 2^M kinds of different results of $\sum_{m=1}^{M} A_m X_m$. If we construct a LUT which can store all the possible combination of values, we can calculate the value of 2^M in advance and store them in the LUT. Using X_{mn} as the LUT address signal, the shifting (2^{-1} operation) and adding operation are carried out on the output of the LUT.

Then $\sum_{m=1}^{M} A_m X_m$ can be realized through N-1 cycles and the result of multiplication-accumulation can be achieved directly. So the complicated multiplication-accumulation operation is converted to the shifting and adding operation. The parallel computing is adopted to improve the speed of calculation. The complicated multiplication-accumulation operation is converted to the shifting and adding operation when the DA algorithm is directly applied to realize linear time-invariant system. However, the scale of the LUT will increase exponentially with the coefficient. If the coefficient is small, it is very convenient to realize through the rich structure of FPGA LUT; while the coefficient is large, it will take up a lot of storage resources of FPGA and reduce the calculation speed. Meanwhile, the N-1 cycles also result in the too long LUT time and the low computing speed.

From Eq. 2, X_m can be expressed as Eq.4.

$$X_m = \frac{1}{2}[X_m - (-X_m)] \text{---------- Eq.4}$$

Where the $-X_m$ can be expressed as Eq. Eq.5 according to the binary complement operation.

$-X_m = -\overline{x_{m0}} + \sum_{n=1}^{N-1} \overline{x_{mn}}\, 2^{-n} + 2^{-(N-1)}$-Eq .5

Put Eq. (5) and Eq. (2) into Eq. (4), Eq. (6) can be achieved.

$$X_m = \frac{1}{2}\left[-(x_{m0} - \overline{x_{m0}}) + \sum_{n=1}^{N-1}(x_{mn} - \overline{x_{mn}})2^{-n} - 2^{-(N-1)}\right] - - - - - \text{Eq. 6}$$

For convenience, two variables are defined as follows:

$$\varphi_{m0} = -(x_{m0} - \overline{x_{m0}})$$
$$\varphi_{mn} = -(x_{mn} - \overline{x_{mn}})$$

In which the, as the value of x_{mn} is 0 or 1, so the value of φ_{mn} and φ_{m0} is ±Eq.6 can be expressed as Eq.7

$X_m = \frac{1}{2}\left[\sum_{n=1}^{N-1} \varphi_{mn}\, 2^{-n} - 2^{-(N-1)}\right] - - Eq.7$

Put Eq. 7 in to Eq.1, Eq.8 can be achieved as follows

$$Y = \frac{1}{2}\sum_{k=1}^{K} A_k \left[\sum_{n=1}^{N-1} \varphi_{mn}\, 2^{-n} - 2^{-(N-1)}\right]$$

$\frac{1}{2}\left[\sum_{n=1}^{N-1}\sum_{m=1}^{M} A_m\, \varphi_{mn}\, 2^{-n} - \sum_{m=1}^{M} A_m 2^{-(N-1)}\right]$ ---------- Eq.8

As there are 2^M different kinds of results of $\sum_{m=1}^{M} A_m\, \varphi_{mn}$ and the value of φ_{mn} is ±1, so the results show positive and negative symmetry property. If the positive and negative sign are not considered, there are only 2^{M-1} different kind of results and the size of storage will reduce by half.

Through the algorithm optimization, Eq.8 can be simplified as Eq.9.

$Y = \frac{1}{2}\left[\sum_{n=1}^{N-1}\sum_{m=1}^{M/2} A_m\, x_{mn}\, 2^{-n} - \sum_{m=1}^{M/2} A_m 2^{-(N-1)}\right]$----------Eq.9

Although the storage size reduces by half through the algorithm optimization, but the scale of the LUT will increase exponentially with the "m".so the scale of the LUT can be further reduced by decreasing the size of "m" then the

$$\sum_{m=1}^{M/2} A_m\, x_{mn}$$

in Eq.9 can be defined as Eq. 10.

$$\sum_{m=1}^{M/2} A_m x_{mn} = \sum_{m=1}^{a} A_m x_{mn} + \sum_{m=a+1}^{b} A_m x_{mn} \ldots\ldots\ldots + \sum_{m=y}^{z} A_m x_n - Eq.10$$

In which, $z \geq y$, $y \geq b$, $b \geq a+1$,$a > 1$,so an inner product operation with the scale of 2^M will be realized through several LUTs with different or same depth and adders. The scale of the memory is $2^a+2^{b-a}+\ldots..+2^{z-y}+2^{M/2-z}$

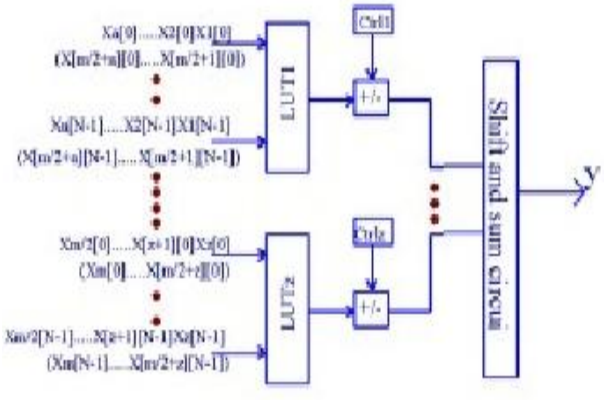

Figure 9 The structure through the algorithm improvement

Though the algorithm improvement, the hardware resource is reduced and the operation speed is improved. The simplified hardware circuit structure is shown in Fig.9. The hardware circuit is shown in Fig.10.

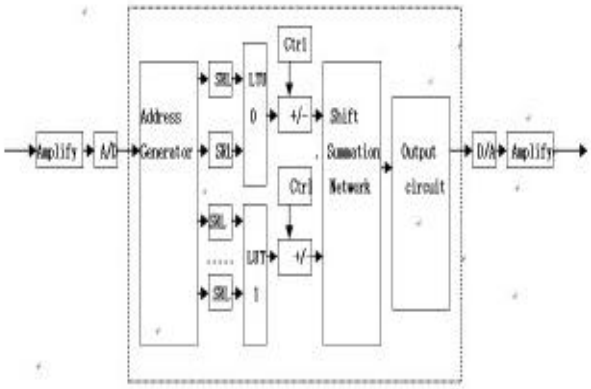

Figure 10 The circuit structure of FIR system

When using the DA algorithm to implement the linear time-invariant system. The pre-storing value corresponding to the upper half of the memory address of LUT storage will be the negative of the lower half and the LUT reduces by half using symmetry. The address maker circuit generates the LUT address. The upper half of the address looks up its corresponding pre-storing value. Meanwhile, the address is used as control-adding-decrease implement to complete the positive and negative conversion between the pre-storing value corresponding to the upper and lower half of it. According to result of the improvement and optimization, the LUT is divided into two 4-input LUTs and the address maker circuit divides the input signals into four segments in accordance with the 4-input LUT. The speed of signal sampling under the control of the FPGA can be adjusted. The data buffer can be established according to the order of the filter. As the designed filter is a 16th-order one, so the sampled serial data can be sent to the 20 bits serial-in- parallel-out shift register, and then the data is divided and sent to the LUT in turn. As the coefficient is amplified 216 times, the obtained result is reduced by the output circuit accordingly.

IV. Results and Conclusions

The complicated multiplication-accumulation operation is converted to the shifting and adding operation when the DA algorithm is directly applied to realize FIR filter. Aiming at the problems of the best configuration in the coefficient of FIR filter, the storage resource and the calculating speed, the DA algorithm is optimized and improved in the algorithm structure, the memory size and the look-up table speed. The arithmetic expression has clear layers of derivation process and the circuit structure is reasonable, which make the memory size smaller and the operation speed faster. The design improves greatly compared to the conventional FPGA realization and it can be flexibility applied to implement high-pass, low-pass and band-stop filters by changing the order and the LUT coefficient. The proposed new structures exploit the nature of symmetric coefficients of odd length and further reduce the amount of multipliers required at the expense of additional adders. Since

multipliers outweigh adders in hardware cost, it is profitable to exchange multipliers with adders. Moreover, the number of increased adders stays still when the length of FIR filter becomes large, whereas the number of reduced multipliers increases along with the length of the FIR filter.

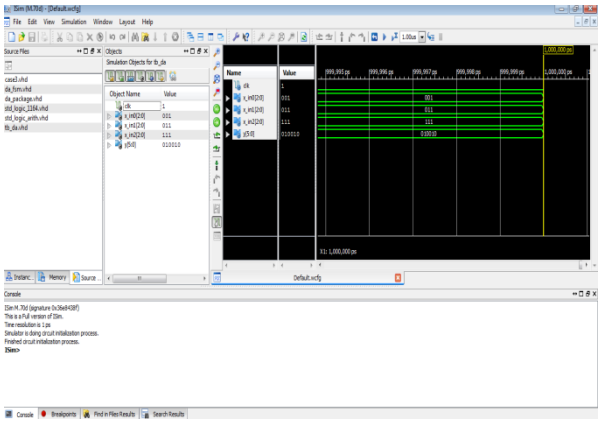

Figure 11 Simulation Result for DA Algorithm

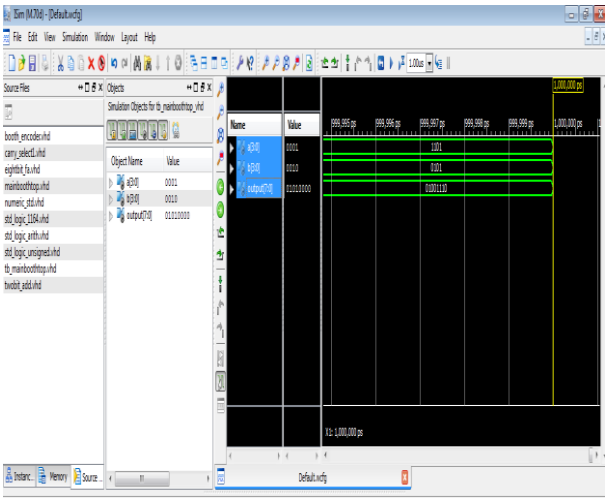

Figure 12 Simulation Results of FIR Result

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

References

[1] K. K. Parhi, VLSI Digital Signal Processing systems: Design and Implementation. New York: Wiley, 1999.
[2] Z.-J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," IEEE Trans. Signal Process., vol. 39, no. 6, pp. 1322–1332, Jun. 1991.
[3] J. I. Acha, "Computational structures for fast implementation of L-path and L-block digital filters," IEEE Trans. Circuits Syst., vol. 36, no. 6, pp. 805–812, Jun. 1989.
[4] C. Cheng and K. K. Parhi, "Hardware efficient fast parallel FIR filter structures based on iterated short convolution," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 8, pp. 1492–1500, Aug. 2004.
[5] C. Cheng and K. K. Parhi, "Furthur complexity reduction of parallel FIR filters," in Proc. IEEE ISCAS, May 2005, vol. 2, pp. 1835–1838.
[6] C. Cheng and K. K. Parhi, "Low-cost parallel FIR structures with 2-stage parallelism," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 2, pp. 280–290, Feb. 2007.
[7] I.-S. Lin and S. K. Mitra, "Overlapped block digital filtering," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 8, pp. 586–596, Aug. 1996.
[8] Y.-C. Tsao and K. Choi, "Area-efficient parallel FIR digital filter structures for symmetric convolutions based on fast FIR algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 2, pp. 366–371, Feb. 2010.
[9] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," J. VLSI Signal Process. Syst., vol. 17, no. 1, pp. 75–92, Sep. 1997.
[10] J. G. Chung and K. K. Parhi, "Frequency-spectrum-based low-area lowpower parallel FIR filter design," EURASIP J. Appl. Signal Process., vol. 2002, no. 9, pp. 444–453, Jan. 2002.