IJCAES
ISSN: 2231-4946
Volume III, Special Issue, August 2013

International Journal of Computer Applications in Engineering Sciences
Special Issue on National Conference on Information and Communication (NCIC'13)
www.caesjournals.org

LUT Optimization Using Combined APC-OMS Technique For Memory-Based Computation

Mrs.R.RAMYA[#1] Mrs.S.SUDHA[#2]
[#1]*M.E. VLSI DESIGN, Easwari Engineering College,Ramapuram.*
[#2]*Faculty of ECE Dept, Easwari Engineering College, Ramapuram.*
[#1]rramyabe@gmail.com , [#2]sudha.s@srmeaswari.ac.in

Abstract: The efficient memory computing system alternative to conventional logic computing is required in DSP applications. The LUTdesigned using combined APC-OMS technique can be used for these applications. APC-OMS technique reduces the LUT size to one-fourth of its original. In this paper, FIR filter is designed using conventional LUT. If conventional LUT is used, the on-chip memory size is getting larger. The memory size can be reduced by decomposing the LUT. FIR filter is designed using multiplexer which is used to select the filter co-efficients. With LUT, area and delay efficiency cannot be achieved.This design has less number of memory accesses and area than LUT based FIR design. By the proposed method, 50% area efficiency and 20%throughput is achieved. The filter designs are simulated using Modelsim6.4a and are synthesized using Quartus II 9.0.

Keywords: Look-up table(LUT), Anti-symmetric Product Coding(APC), Odd Multiple Storage(OMS), Finite Impulse Response(FIR).

I. INTRODUCTION

Finite impulse response (FIR) filters are common components in many digital signal processing (DSP) systems [5-10]. With the increasingly development in very large scale integration (VLSI) technology, the FIR filter with less hardware requirement and less latency has become more and more important for real time applications. Because the complexity of implementation grows with the length of filter, several algorithms have been made to develop effective architectures for realization of FIR filters in application specific integrated circuits (ASIC) and field programmable gate arrays (FPGA) platforms and one of them is distributed arithmetic (DA). The main portion of DA-based computation is lookup table (LUT) that stores the pre-computed values and can be read out easily, which makes DA-based computation well suitable for FPGA realization, because the LUTs are the basic components of FPGA.

Moreover, this technology has a number of attractive features such as simplicity, regularity and modularity of architecture. Also, the DA technique can be designed to meet various speed requirements like high-speed and medium speed implementations. In high speed, all bits of one word are processed per clock. In medium-speed implementation, several bits of one word (not all bits) are processed per clock. In recent years, DA-based FIR filter has gained substantial popularity as a primary DSP operation and are rapidly replacing classic analog filters.

The N-Tap FIR filter can be described as:

$y=<h,x>= \sum_{n=0}^{N-1} h[n]x[n]$ ---------(1)

Where h[n] is the filter coefficient and x[n] is the input sequence to be processed. The FIR structure consists of a series of multiplication and addition units, and consume N MAC blocks of FPGA, which are expensive in high speed system. Comparing to traditional direct arithmetic, Distributed Arithmetic will save considerable hardware resources by using LUT to take the place of MAC units [5]. Another virtue of this method is that it will not reduce system speed with the increase of the input data bit width or the filter coefficient bit width, which will occur in traditional direct method which consume considerable hardware resources.

The "impulse response" of an FIR filter is the set of FIR coefficients. By Putting an impulseinto a FIR filter which consists of a "1" sample followed by many "0" samples, then the output of the filterwill be the set of coefficients, as the 1 sample moves through past each coefficient in turn to form the output.
Tap - A FIR tap is a coefficient and delay pair. The number of FIR taps, designated as "N" which is anindication of the amount of memory, number of calculations required to implement the filter, and also the amount of "filtering" the filter can be able to do. If number of taps is more, then stop-band attenuation will be more.

Multiply-Accumulate (MAC) - In a FIR context type, a MAC is the operation of multiplying a coefficient with the corresponding delayed data sample and then accumulating the result. FIR filters generally requiresone MAC per tap. Most of the DSP microprocessors implements the MAC operation in a single instruction.

II.PROPOSED SCHEMES:

1.Conventional LUT:

A conventional look-up table (LUT)-based multiplier is shown in Fig. 1,where A is a fixed coefficient, and X is an input word to be multiplied with A. Assuming X to be positive binary number of word length L, there can be 2^L possible values of X, and accordingly, there can be 2^L values of product C = A.X. therefore, for memory-based multiplication, an LUT of 2^L words, consisting of precomputed product values corresponding to all possible values of values of X, is conventionally used. The product word A . X_i is available as its output. Several architectures have been reported in the literature for memory-based implementation of DSP algorithms involving orthogonal transforms and digital filters. However, there is no significant work on LUT optimization for memory-based multiplication

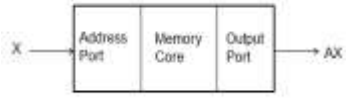

Fig.1. Conventional LUT-based Multiplier

In the new approach of LUT design, only the odd multiples of the fixed coefficient are required to be stored , which we have referred to as the *odd-multiple-storage* (OMS) scheme. In addition, it is also shown that, by the *antisymmetric product coding* (APC) approach, the LUT size can also be reduced to half, where the product words are recoded as anti-symmetric pairs. The APC approach, although providing a reduction in LUT size by a factor of two, incorporates substantial overhead of area and time to perform the two's complement operation of LUT output for sign modification and that of the input operand for input mapping.

When the APC approach is combined with the OMS technique, the two's complement operations could be very much simplified since the input address and LUT output could always be transformed into odd integers. However, the OMS technique cannot be combined with the APC scheme, since the APC words generated are odd numbers. Moreover, the OMS scheme in does not provide an efficient implementation when combined with the APC technique. In this brief, we therefore present a different form of APC and combined that with a modified form of the OMS scheme for efficient memory based multiplication.

2. LUT Multiplier Using Modified OMS And APC Technique:

The implementation of the proposed APC-OMS combined LUT for memory based multiplier uses two techniques, APC and OMS method is shown in Fig.2.. This method is supposed to reduce the area to one fourth.

This multiplier uses four blocks. The first block converts the given input to address d0d1d2d3, which is produced by combining both the APC and OMS method. The second block converts the address d0d1d2d3 to LUT address. The third block is an stored LUT and fourth block converts the LUT output to the desired output.

The proposed APC–OMS combined design of the LUT for $L = 5$ and for any coefficient width W is shown in Fig.4.1. It consists of an LUT of nine words of $(W + 4)$-bit width, a four-to-nine-line address decoder, a barrel shifter, an address generation circuit, and a control circuit for generating the RESET signal and control word ($s1s0$) for the barrel shifter. The precomputed values of $A \times (2i + 1)$ are stored as $.P_i$ for $i = 0, 1, 2, \ldots, 7$, at the eight consecutive locations of the memory array, as specified in Table 3.2, while $2A$ is stored for input $X = (00000)$ at LUT address "1000," as specified in Table 1.

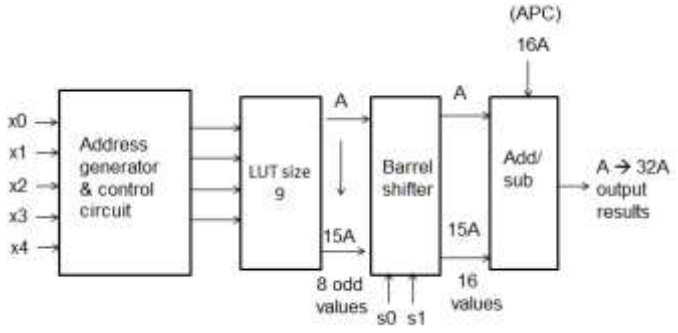

Fig.2. APC-OMS based LUT multiplier

The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., *{wi,* for $0 \le i \le 8$*}*, to select the referenced word from the LUT.
The control bits $s0$ and $s1$ to be used by the barrel shifter to produce the desired number of shifts of the LUT output are generated by the control circuit , according to the relations

$s_0 = (x_0+(x_1+x_4')')'$
$s_1 = (x_0 + x_1)'$

Note that ($s1s0$) is a 2-bit binary equivalent of the required number of shifts specified. The RESET signal can alternatively be generated as ($d3$ AND $x4$). The address-generator circuit receives the 5-bit input operand X and maps that onto the 4-bit address word ($d3d2d1d0$).
In Table 1,it is giventhat, the eight odd multiples, $A \times (2i + 1)$ are stored as Pi, for $i = 0, 1, 2, \ldots, 7$ at eight locations. The even multiples $2A$, $4A$, and $8A$ are derived by left-shift operations of A. In similar way, $6A$ and $12A$ are derived by left shifting $3A$, while $10A$ and $14A$ are derived by left shifting $5A$ and $7A$, respectively. A barrel shifter for producing a maximum of three left shifts could be used to derive all the even multiples of A.
The word to be stored for $X = (00000)$ is not 0, it is supposed to be $16A$, which can be obtained from A by four left shifts using a barrel shifter. However, if $16A$ is not derived from A, only a maximum of three left shifts is required to obtain all other even multiples of A.
A maximum of three bit shifts can be implemented by a two-stage logarithmic barrel shifter, but the implementation of four shifts requires a three-stage barrel shifter. Therefore, if we store $2A$ at (1000), the product $16A$ can be derived by three arithmetic left shifts which is required at location X=(00000). The product values and encoded words for input words $X = (00000)$ and (10000) are shown in Table 1. For $X = (00000)$, the desired encoded word $16A$ is derived by 3-bit left shifts of $2A$ which is stored at address (1000). For $X = (10000)$, the APC word "0" is derived by resetting the LUT output, by an active-high RESET signal given by

RESET = $\sim(x_0+x_1+x_2+x_3) .x_4$

Input X' $x_3'x_2'x_1'x_0'$	Product values	# of shifts	Shifted input X''	Stored APC words	Address $d_3d_2d_1d_0$
0 0 0 1	A	0	0001	P0= A	0000
0 0 1 0	2 x A	1			

0 1 0 0	4 x A	2			
1 0 0 0	8 x A	3			
0 0 1 1	3A	0	0011	P1=3A	0001
0 1 1 0	2 x 3A	1			
1 1 0 0	4 x 3A	2			
0 1 0 1	5A	0	0101	P2=5A	0010
1 0 1 0	2 x 5A	1			
0 1 1 1	7A	0	0111	P3=7A	0011
1 1 1 0	2 x 7A	1			
1 0 0 1	9A	0	1001	P4=9A	0100
1 0 1 1	11A	0	1011	P5=11A	0101
1 1 0 1	13A	0	1101	P6=13A	0110
1 1 1 1	15A	0	1111	P7=15A	0111

Input X $x_4x_3x_2x_1x_0$	Product values	Encoded word	Stored values	# of shifts	Address $d_3d_2d_1d_0$
1 0 0 0 0	16A	0	---	---	---
0 0 0 0 0	0	16A	2A	3	1 0 0 0

TABLE. 1. OMS-based design of LUT of APC words

The 5-bit input word *X* can be mapped into a 4-bit LUT address ($d_3d_2d_1d_0$), by the mapping relations,

$d_i = x''_{i+1}$, for i = 0, 1, 2 and $d_3 = \sim(x''_0)$

Where $X'' = (x''_3\, x''_2\, x''_1\, x''_0)$ is generated by shifting-out all the leading zeros of X' by an arithmetic right shift followed by address mapping,

$X'' = \{Y_L$, if x_4=1
Y'_L, if x_4=0

where Y_L and Y'_L are derived by circularly shifting-out all the leading zeros of X_L and X'_L respectively.

3. FIR Implementation with conventional LUT:

The implementation of FIR filters on FPGA based on traditional method costs considerable hardware resources, which goes against the decrease of circuit scale and the increase of system speed. A new design and implementation of FIR filters using Distributed Arithmetic is provided to solve this problem. Distributed Arithmetic structure is used to increase the resource usage while pipeline structure is also used to increase the system speed. In addition, the divided LUT method is also used to decrease the required memory units.

Here is the proposed structure for memory-based realization of an N -tap FIR filter, and discussed the design of memory cell to be used as LUT in the structure. The input-output relationship of an N -tap FIR filter in time-domain is given by

$$y(n)=h(0).x(n) +h(1).x(n-1) +h(2).x(n-2) +\ldots\ldots + h(N-1).x(n-N+1)$$

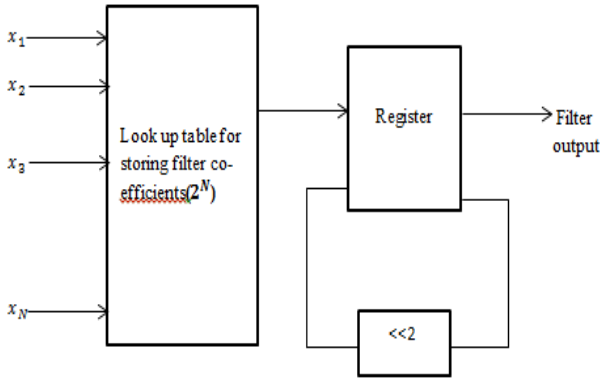

Fig.3. LUT based design of FIR Filter

The LUT based design of FIR filter is shown in Fig 3. It reduces the logic needed to implement MAC operator to only adders and shifters. It can be implemented on "Look−up−table" based FPGAs without wasting resources. It has several implementation based on several trade-offs. Most FIR filters have constant operators so using this algorithm they can be implemented with very custom circuits. The advantage of this approach is its efficiency of mechanization. Here the look-up table is used to store the filter co-efficients. The input samples will address the co-efficients stored in LUT. For an N-Tap filter, 2^N co-efficients will be stored in LUT. The output of FIR filter will be found by shifting the co-efficients based on position of input sample bits. Here conventional LUT is used to store the co-efficients of all the combinations. Here memory usage for simulation will be increased when number of taps increases.

4.FIR Implementation with decomposed LUT:

FIR filters can be designed with decomposed architecture which reduces the memory usage when comparing to conventional LUT for storing co-efficients shown in Fig 4. The input samples will be divided and will be given to decomposed LUT based on its position. The N-Tap filetr can be designed with many number of LUTs. Increasing the number of LUTs which will reduces the memory usage. In turn, the on-chip memory size is getting reduced.

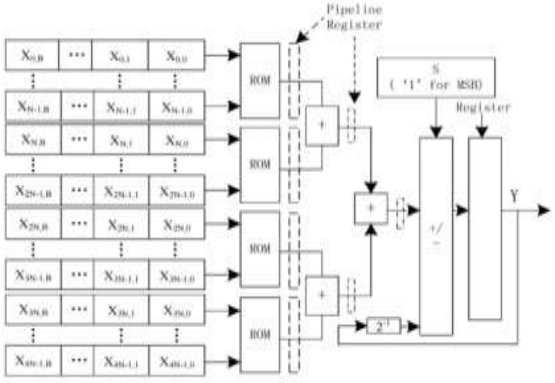

Fig.4. FIR with Decomposed LUTs.

5.FIR filter Design using Multiplexer:

FIR filter is designed using multiplexer for addressing signal co-efficients which is shown in Fig.5. The 2:1 Multiplexer is used to select the filter co-efficients based on input signal samples. The input sample values will be found out from Matlab. The floating point values will be converted into binary values. Those values will be given as input samples. For each sample, one multiplexer will be used. This design is very efficient than LUT design in terms of area and delay. Only N- multiplexers will be used for N-tap filter. Carry Look ahead adder is used to add the filter co-efficients. The carry look ahead adder is having less propagation delay than Ripple carry adder. The output will

be stored in Register and it will shift the values to get into corresponding bit position. The signal values are floating point values, so right shifter is used to shift the values.

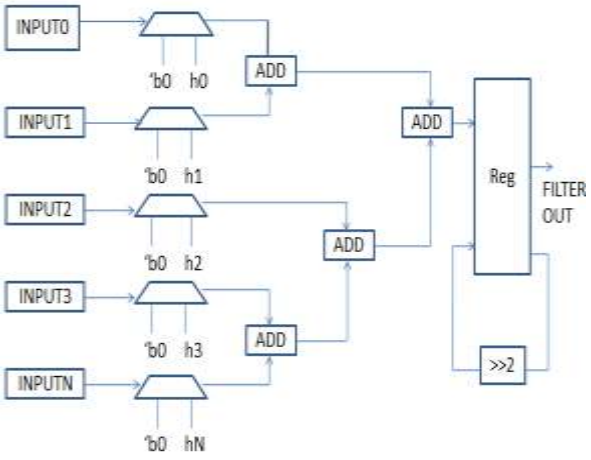

Fig.5. Multiplexer based FIR Filter Design

III.RESULTS AND DISCUSSION

The design is coded in VerilogHDL and simulated using Modelsim 6.4a and synthesized using Quartus II 9.0. Quartus II is used to analyse the logic elements used for conventional LUT-based multiplier and APC-OMS based LUT multiplier. In APC-OMS based multiplier, the 5-bit input is given to address generation circuit. It generates address to address LUT, select lines for barrel shifter and reset values for LUT. The 4-bit address will be given to 4-to-9 decoder to map for LUT address. The stored LUT is stored with eight odd multiples and one even multiples in nine locations. One of this value will be addressed from decoder and will be given to barrel shifter. According to select lines, the values will be shifted and given to ADD/SUB unit to get desired output. It is also designed for signed and unsigned operands. This LUT design is also used to implement for higher input sizes by decomposing into several 5-bit LUTs. The area utilization for APC-OMS based LUT is less than conventional type LUT. The simulation output for LUT multiplier is shown in Fig. 6.

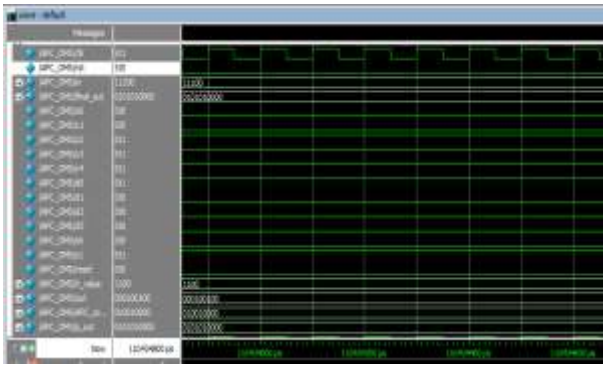

Fig.6.APC-OMS based LUT Multiplier.

FIR design is coded in Verilog HDL, simulated in ModelSim and synthesized in QuartusII. The output of FIR filter using conventional LUT is shown in Fig.7. The input samples are convoluted with filter co-efficients to get filter output. The co-efficients are stored in LUT which are to be addressed by input samples.

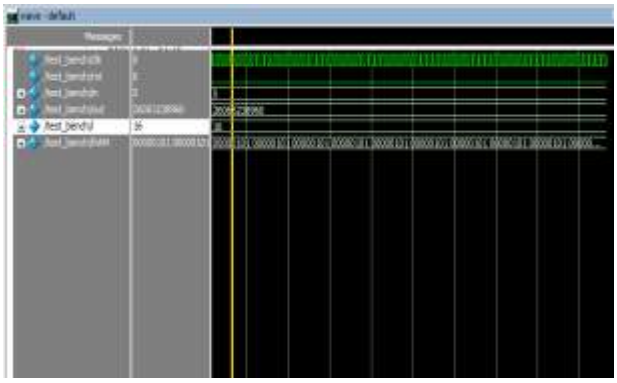

Fig.7. FIR filter with conventional LUT

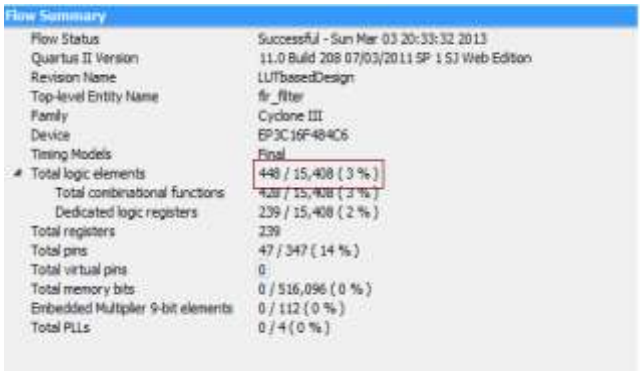

Fig.8.Synthesis report of LUT based FIR design

The multiplexer based FIR filter output is shown in Fig.9. The number of multiplexers will be increased if tap increases. For each bit position of all the input samples, the filter co-efficients will be selected and will be added together and stored in Register. Based on sample position, the output will be shifted and added together with previous position outputs. The synthesis report is shown in Fig.10.

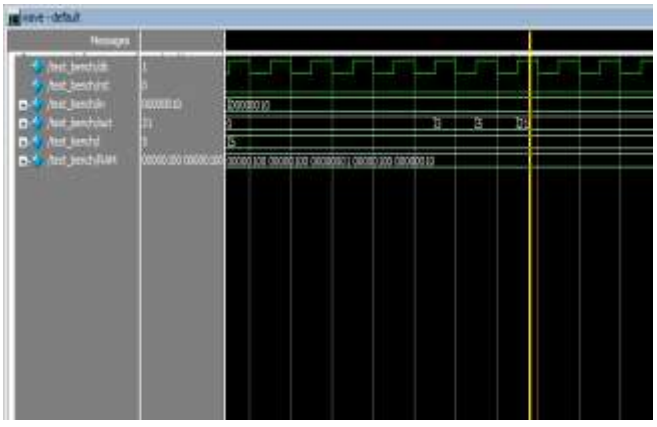

Fig.9. FIR Filter using Multiplexer.

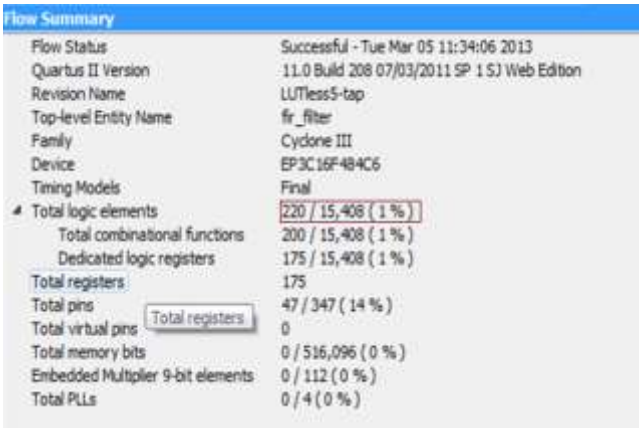

Fig.10.Synthesis Report of Multiplexer Basd FIR design

Comparison of area and delay:

When implementing FIR filter using multiplexer, the area and delay for the design is less comparing to LUT based FIR design. The comparison of area and delay is shown in Table.2.

Parameter	**LUT based FIR design**	**Multiplexer based FIR design**
Area utilization(Logic elements)	**448**	**220**
Delay	**6.652ns**	**5.265ns**

Table.2. Comparison of area and delay.

IV.CONCLUSION

The proposed architecture applies to the main concept of the basic DA technique in implementing the MAC unit and at the same time has many advantages over its basic architecture. The results obtained show that with the proposed architecture, the computation time and the area used is reduced. As the Mux based FIR design has less number of logic elements and memory accesses, it has reduced area and time complexities and achieves better efficiency than LUT based FIR design. The proposed architecture can be used to implement high order FIR filters with different coefficients wordlength without suffering from large LUT construction and with low hardware complexity. The future work is to perform a VLSI implementation of pulse shaping FIR filter for ultra wideband communications.

References:

[1] P. K. Meher, "Systolic designs for DCT using a low-complexity concurrent convolutional formulation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 9, pp. 1041–1050, Sep. 2006.

[2] P. K. Meher, "Memory-based hardware for resource-constrained digital signal processing systems," in *Proc. 6th Int. Conf. ICICS*, Dec. 2007, pp. 1–4.
[3] P. K. Meher, "New approach to LUT implementation and accumulation for memory-based multiplication," in *Proc. IEEE ISCAS*, May 2009, pp. 453–456.
[4] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proc. ISIC*, Dec. 2009, pp. 663–666.
[5] D.J. Allred,H.Yoo, V. Krishnan, W. Huang, and D.V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no.7,pp. 1327-1337, Jul. 2005.
[6] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D.V. Anderson,"A novel high performance distributed arithmetic adaptive filter implementation on an FPGA," *in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2004, vol. 5, pp. V-161-C-164.
[7] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D.V. Anderson,"An FPGA implementation for a high throughput adaptive filter using distributed arithmetic," in *Proc. 12th Annu. IEEE Symp. Field-Programmable Custom Comput. Mach.*, 2004, pp. 324-325.
[8] K.Kammeyer,"Digital Filter realization in distributed arithmetic," in Proc. Eur. Conf. Circuit Theory Des., Genoa, Italy.
[9] M. A. M. Eshtawie and M. Othman, "On-line DA-LUT architecture for high-speed high-order digital FIR filters," *in Proceedings of the IEEE International Conference on Communication Systems (ICCS)*, Singapore, November. 2006.
[10] Kim, Kyung- Sheng, Kwyro Lee, "Low-power and area efficient FIR filter implementation suitable for multiple tape," *IEEE Trans. on VLSI Systems* , February 2003.