ISSN:0975-9646

Implementation of Area Efficient Memory-Based FIR Digital Filter Using LUT-Multiplier

K.Purnima, S.AdiLakshmi, M.Jyothi

Department of ECE, K L University
Vijayawada, INDIA

***Abstract*— Memory based structures are well-suited for many digital signal processing (DSP) applications, which involve multiplication with a fixed set of coefficients. Memory-based structures are more regular compared with the multiply-accumulate Structures and have many other advantages of less area and reduced latency implementation since the memory-access-time is much shorter than the usual multiplication-time compared to the conventional multipliers. Distributed arithmetic (DA)-based computation is popular for its potential for efficient memory-based implementation of finite impulse response (FIR) filter. In this paper, however, we show that the look-up-table (LUT)-multiplier-based approach, where the memory elements store all the possible values of products of the filter coefficients could be an area-efficient alternative to DA-based design of FIR filter with the same throughput of implementation. By operand and inner-product decompositions, respectively, we have designed the conventional LUT-multiplier-based and DA-based structures for FIR filter of equivalent throughput, where the LUT-multiplier-based design involves nearly the same memory and the same number of adders, and less number of input register at the cost of slightly higher adder-widths than the other. Moreover, we present new approach to LUT-based multiplication, which could be used to reduce the memory size to half of the conventional LUT-based multiplication. Besides, we present a modified transposed form FIR filter, where a single segmented memory-core with only one pair of decoders are used to minimize the combinational area. We have implemented the FIR filter using proposed LUT-multiplier and LUT-multiplier based transposed form FIR filter both of order four using Xilinx tool in VHDL.**

***Keywords*— DSP chip, memory-based structures, distributed arithmetic, LUT-based multiplier, FIR filter, Xilinx tool, VLSI.**

I. INTRODUCTION

Finite-Impulse response (FIR) digital filter is widely used as a basic tool in various signal and image processing applications [1]. Many applications in digital communication, speech processing, seismic signal processing and several other areas require large order FIR filters [2]. Since the number of multiply-accumulate (MAC) operations required per filter output increases linearly with the filter order; real-time implementation of these filters of large orders is a challenging task. Therefore, several attempts have been made and continued to develop low-complexity dedicated VLSI systems for these filters [3]-[5].

II. MEMORY-BASED STRUCTURES

In this paper, we use the phrase "*memory-based structures*" or "*memory-based systems*" for those systems where memory elements like RAM or ROM is used either as a part or whole of an arithmetic unit [10]. Memory-based structures are more regular compared with the multiply-accumulate structures; and have many other advantages, e.g., greater potential for high-throughput and reduced-latency implementation, (since the memory-access-time is much shorter than the usual multiplication-time) and are expected to have less dynamic power consumption due to less switching activities for memory-read operations compared to the conventional multipliers. Memory-based structures are well-suited for many digital signal processing (DSP) algorithms, which involve multiplication with a fixed set of coefficients.

There are two basic variants of memory-based techniques. One of them is based on distributed arithmetic (DA) for inner product computation [11]-[13], [14], [15] and the other is based on the computation of multiplication by look-up-table (LUT). In the LUT-multiplier-based approach, multiplications of input values with a fixed-coefficient are performed by an LUT consisting of all possible pre-computed product values corresponding to all possible values of input multiplicand, while in the DA-based approach, an LUT is used to store all possible values of inner-products of a fixed –N-point vector with any possible N-point bit-vector. If the inner-products are implemented in a straight-forward way, the memory-size of LUT-multiplier based implementation increases exponentially with the word length of input values, while that of the DA-based approach increases exponentially with the inner-product-length. Attempts have been made to reduce the memory-space in DA-based architectures using offset binary coding (OBC) [11] and group distributed technique [13]. A decomposition scheme is suggested in a recent paper [14] for reducing the memory-size of DA-based implementation of FIR filter. But, it is observed that the reduction of memory-size achieved by such decompositions is accompanied by increase in latency as well as the number of adders and latches.

III. LUT DESIGN FOR MEMORY BASED MULTIPLICATION

The basic principle of memory-based multiplication is depicted in Fig. 1. Let A be a fixed coefficient and X be an input word to be multiplied with A. If we assume X to be an unsigned binary number of word-length L, there can be 2^L possible values of X, and accordingly, there can be 2^L possible values of product C=A.X . Therefore, for the conventional implementation of memory-based multiplication [15], a memory unit of 2^L words is required to be used as look-up-table consisting of pre-computed product values corresponding to all possible values of X. The product-word, $(A.X_i)$ for $0 \leq X_i \leq 2^L-1$, is stored at the memory location whose address is the same X_i as the binary value of X_i , such that if L-bit binary value of X_i is used as address for the memory-unit, then the corresponding product value is read-out from the memory.

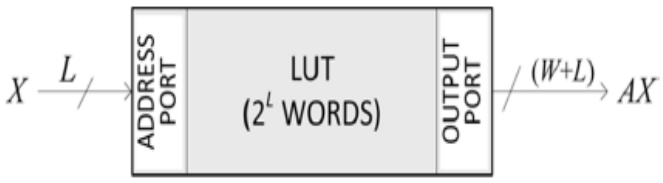

Fig.1. Conventional Memory-Based Multiplier

IV. DISTRIBUTED ARITHMETIC ARCHITECTURE

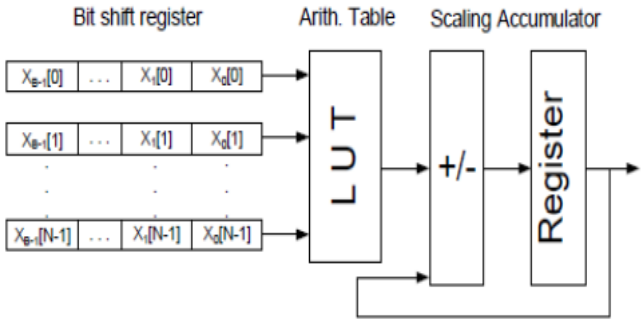

Fig.2. Distributed arithmetic block diagram

DA is a bit-serial operation that implements a series of fixed-point MAC operations in a fixed number of steps, regardless of the number of terms to be calculated. One problem with original DA architecture is that its LUT size (2*K*-words) grows exponentially as the filter order *N* increase.

If the inner-products are implemented in a straight-forward way, the memory-size of DA based implementation increases exponentially with the inner-product-length. Attempts have been made to reduce the memory-space in DA-based architectures for reducing the memory-size of DA-based implementation of FIR filter. But, it is observed that the reduction of memory-size achieved by such decomposition is accompanied by increase in latency as well as the number of adders and latches.

V. MEMORY-BASED FIR FILTER USING CONVENTIONAL LUT

The recursive computation of FIR filter output can also be understood from the FIR filter structure using conventional LUT-multiplier as shown in Fig. 3.1. Each multiplication node performs the multiplication of an input sample value with the absolute value of a filter coefficient. The AS node adds or subtracts its input from top with or from that of its input from the left when the corresponding filter coefficient is positive or negative, respectively. It may be noted here that each of the multiplication nodes performs multiplications of input samples with a fixed positive number.

This feature can be utilized to implement the multiplications by an LUT that stores the results of multiplications' of all possible input values with the multiplying coefficient of a node as unsigned numbers. The multiplication of an L-bit unsigned input with W-bit magnitude part of fixed filter-weight, to be performed by each of the multiplication-nodes of the DFG, can be implemented conventionally by a dual-port memory consisting of words of (W+L) bit width. Each of the nodes of the DFG along with a neighbouring delay element can be mapped to an add-subtract (AS) cell. A fully pipelined structure for N-tap FIR filter for input word length L=8 is derived accordingly from the DFG. It consists of N memory-units for conventional LUT-based multiplication, along with (N-1) AS cells and a delay register. All the 8 bits of current input sample x (n) are fed to all the LUT-multipliers in parallel as a pair of 4-bit addresses X1 and X2 and the structure of the LUT-multiplier is shown in Fig 3.2.

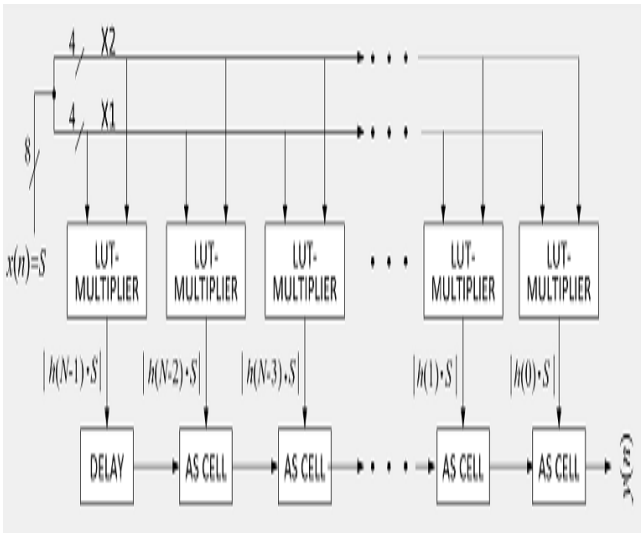

Fig.3.1. Conventional LUT-multiplier-based structure of an n-tap FIR Filter for input-width L=8.

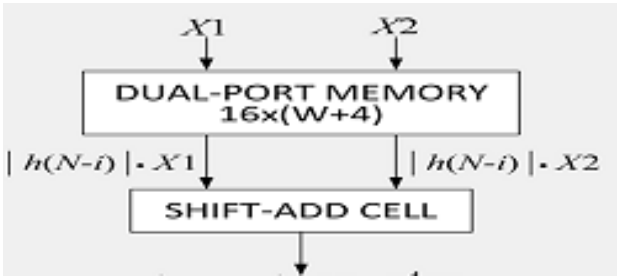

Fig.3.2. Structure of each LUT-multiplier

It consists of a dual-port memory unit of size [16 x (W+4)] (consisting of 16 words of (W+4)-bit width) and a shift-add (SA) cell. The SA cell shifts its right-input to left by four bit-locations and adds the shifted value with its other input to produce a (W+8)-bit output. The shift operation in the shift-add cells is hardwired with the adders, so that no additional shifters are required. The outputs of the multipliers are fed to the pipeline of AS cells in parallel. Each AS cell performs exactly the same function as that of the AS node of the DFG. It consists of either an adder or a subtracter depending on whether the corresponding filter weight h(n) is positive or negative, respectively. Besides, each of the SA cells consists of a pipeline latch corresponding to the delays in the DFG of Fig 3.1.

VI. MEMORY-BASED FIR FILTER USING PROPOSED LUT MULTIPLIER

The realization of digital FIR filter using proposed LUT based multiplier is done by using direct form realization structure of digital FIR filter. The equation, which defines the FIR filter with output sequence y[n] in terms of its input sequence x[n]:

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k] \qquad (1)$$

Where x[n] is the input signal, y[n] is the output signal, h[k] is the coefficients of FIR filter frequency response, and N is the filter order.

A. Proposed LUT Design Based 4-Bit Multiplier

The proposed LUT-based multiplier for input word-size L=4 is shown in fig.4. It consists of a memory-array of eight words of (W+4)-bit width and a 3-to-8 line address decoder, along with a NOR-cell, a barrel-shifter, a 4-to-3 bit encoder to map the 4-bit input operand to 3-bit LUT-address, and a control circuit for generating the control-word for the barrel-shifter, and the RESET signal for the NOR-cell.

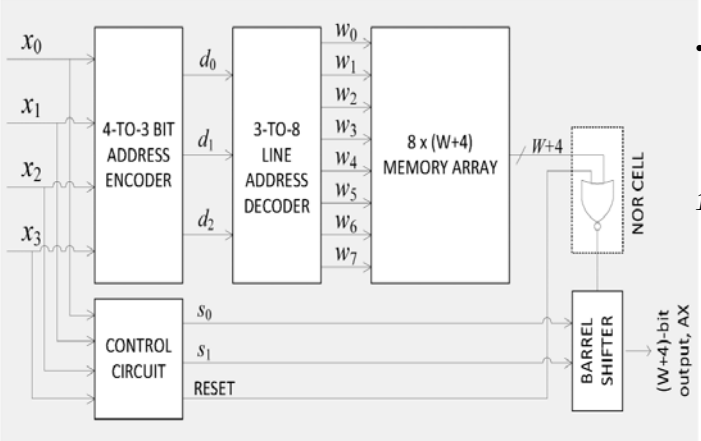

Fig.4. Proposed LUT design based 4-bit multiplier

address $d_2d_1d_0$	word symbol	stored value	input $x_3x_2x_1x_0$	product value	# of shifts	control s_1 s_0
0 0 0	$P0$	A	0 0 0 1	A	0	0 0
			0 0 1 0	$2^1 \times A$	1	0 1
			0 1 0 0	$2^2 \times A$	2	1 0
			1 0 0 0	$2^3 \times A$	3	1 1
0 0 1	$P1$	$3A$	0 0 1 1	$3A$	0	0 0
			0 1 1 0	$2^1 \times 3A$	1	0 1
			1 1 0 0	$2^2 \times 3A$	2	1 0
0 1 0	$P2$	$5A$	0 1 0 1	$5A$	0	0 0
			1 0 1 0	$2^1 \times 5A$	1	0 1
0 1 1	$P3$	$7A$	0 1 1 1	$7A$	0	0 0
			1 1 1 0	$2^1 \times 7A$	1	0 1
1 0 0	$P4$	$9A$	1 0 0 1	$9A$	0	0 0
1 0 1	$P5$	$11A$	1 0 1 1	$11A$	0	0 0
1 1 0	$P6$	$13A$	1 1 0 1	$13A$	0	0 0
1 1 1	$P7$	$15A$	1 1 1 1	$15A$	0	0 0

s_0 and s_1 are control bits of the logarithmic barrel-shifter.

TABLE.1 LUT words and product values for input word length L=4

Although 2^L possible values of X correspond to 2^L possible values of C=A.X, recently we have shown that only (2^L/2) words corresponding to the odd multiples of A may only be stored in the LUT [16]. One of the possible product words is zero, while all the rest (2^L/2)-1 are even multiples of A which could be derived by left-shift operations of one of the odd multiples of A. We illustrate this in Table I for L=4. At eight memory locations, eight odd multiples A x (2i+1) are stored as P_i for i=0, 1, 2… 7. The even multiples 2A, 4A and 8A are derived by left-shift operations of A. Similarly, 6A and 12A are derived by left-shifting 3A, while 10A and 14A are derived by left-shifting 5A and 7A, respectively. The address X= (0000) corresponds to (A.X) =0, which can be obtained by resetting the LUT output. For an input multiplicand of word-size L similarly, only (2^L/2) odd multiple values need to be stored in the memory-core of the LUT, while the other (2^L/2-1) non-zero values could be derived by left-shift operations of the stored values. Based on the above, an LUT for the multiplication of an L-bit input with W-bit coefficient is designed by the following strategy:

- A memory-unit of (2^L/2) words of (W+L)-bit width is used to store all the odd multiples of A.
- A barrel-shifter for producing a maximum of (L-1) left shifts is used to derive all the even multiples of A.
- The L-bit input word is mapped to (L-1)-bit LUT-address by an encoder.
- The control-bits for the barrel-shifter are derived by a control-circuit to perform the necessary shifts of the LUT output. Besides, a RESET signal is generated by the same control circuit to reset the LUT output when X=0.

1) 4-to-3 bits input encoder:

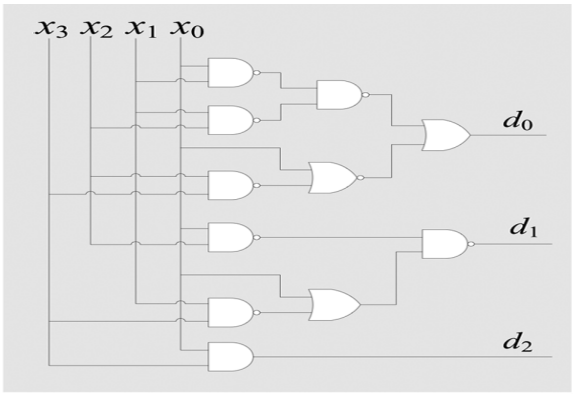

Fig.4.1. 4-to-3 bits input encoder.

The 4-to-3 bit input encoder is shown in Fig.4.1. It receives a four-bit input word and maps that onto the three-bit address word, according to the logical relations

$$d_0 = \overline{(x_0 \cdot x_1)} \cdot \overline{(x_1 \cdot x_2)} \cdot (x_0 + \overline{(x_2 \cdot x_3)}) \qquad 2(a)$$

$$d_1 = \overline{(x_0 \cdot x_2)} \cdot (x_0 + \overline{(x_1 \cdot x_3)}) \qquad 2(b)$$

$$d_2 = x_0 \cdot x_3 \qquad 2(c)$$

The decoder takes the 3-bit address from the input encoder, and generates 8 word-select signals, to select the referenced-word from the memory-array. From Table 1 we find that the LUT output is required to be shifted through 1 location to left when the input operand is one of the values $\{(0\ 0\ 1\ 0), (0\ 1\ 1\ 0), (1\ 0\ 1\ 0), (1\ 1\ 1\ 0)\}$ Two left-shifts are required if is either (0 1 0 0) or (1 1 0 0). Only when the input word X= (1 0 0 0), three shifts are required. For all other possible input operands, no shifts are required. Since the maximum number of left-shifts required on the stored-word is three, a two-stage logarithmic barrel-shifter is adequate to perform the necessary left-shift operations.

2) Control circuit:

The number of shifts required to be performed on the output of the LUT and the control-bits s_0 and s_1 for different values of X are shown in Table I. The control circuit shown in Fig.4.2 accordingly generates the control-bits given by

$$s_0 = \overline{x_0 + \overline{(x_1 + \overline{x_2})}} \qquad 3(a)$$

$$s_1 = \overline{(x_0 + x_1)} \qquad 3(b)$$

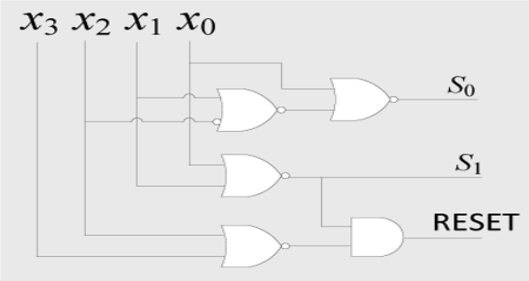

Fig.4.2. Control circuit

The input X= (0 0 0 0) corresponds to multiplication by X=0 which results in the product value equal to 0. Therefore, when the input operand word X= (0 0 0 0), the output of the LUT is required to be reset.

3) Structure of the NOR-cell:

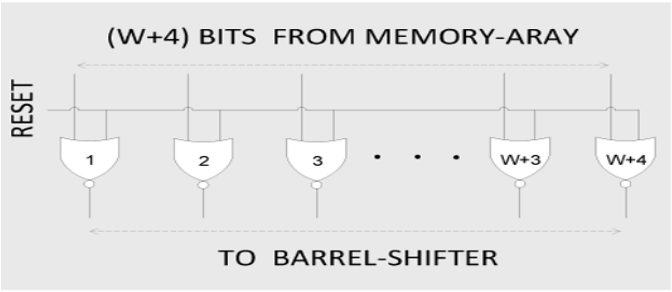

Fig.4.3. Structure of the NOR-cell.

The RESET bit is fed as one of the inputs of all those NOR gates, and the other input lines of (W+4) NOR gates of NOR cell are fed with (W+4) bits of LUT output in parallel. When X= (0 0 0 0), the control circuit in Fig.4.2, generates an active-high RESET according to the logic expression.

$$\text{RESET} = \overline{(x_0 + x_1)} \cdot \overline{(x_2 + x_3)} \qquad (4)$$

4) Two-stage logarithmic barrel-shifter:

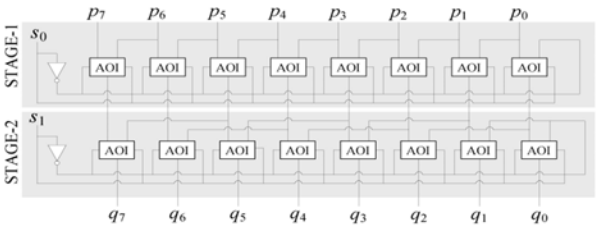

Fig.4.4. Two-stage logarithmic barrel-shifter for W=4

A logarithmic barrel-shifter for W=L=4 is shown in Fig4.4. It consists of two stages of 2-to-1 line bit-level multiplexors with inverted output, where each of the two stages involves (W+4) number of 2-input AND-OR-INVERT (AOI) gates. The control-bits $(s_0, \bar{s}_0)$ and $(s_1, \bar{s}_1)$ are fed to the AOI gates of stage-1 and stage-2 of the barrel-shifter, respectively. Since each stage of the AOI gates perform inverted multiplexing, after two stages of inverted multiplexing, outputs with desired number of shifts are produced by the barrel-shifter in (the usual) un-inverted form.

B. Memory-Based Multiplier Using Dual-Port Memory-Array

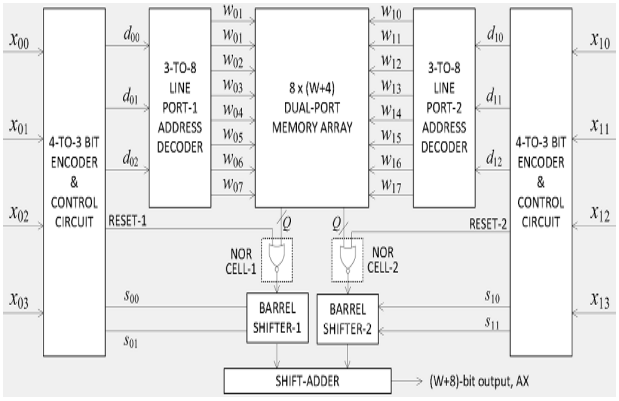

Fig.5. Memory-based multiplier using dual-port memory-array.

Multiplication of an 8-bit input with a w -bit fixed coefficient can be performed through a pair of multiplications using a dual-port memory of 8 words (or two single-port memory units) along with a pair of decoders, encoders, NOR cells and barrel shifters as shown in Fig.5. The shift-adder performs left-shift operation of the output of the barrel-shifter corresponding to more significant half of input by four bit-locations, and adds that to the output of the other barrel-shifter.

C. Memory Based FIR Filter Using Proposed LUT-Multiplier

The memory-based structure of FIR filter (for 8-bit inputs) using the proposed LUT design is shown in Fig. 6.1. It differs from that of the conventional memory-based structure of FIR filter of Fig. 3.1 in two design aspects.

1) The conventional LUT-multiplier is replaced by proposed odd-multiple-storage LUT, so that the multiplication by an L-bit word could be implemented by $(2^{L/2})/2$ words in the LUT in the dual-port memory.

2) Since the same pair of address words X1 and X2 are used by all the N LUT-multipliers in Fig. 3.1, only one memory module with segments could be used instead of N modules. If all the multiplications are implemented by a single memory module, the hardware complexity (used in Fig.3.1) could be eliminated.

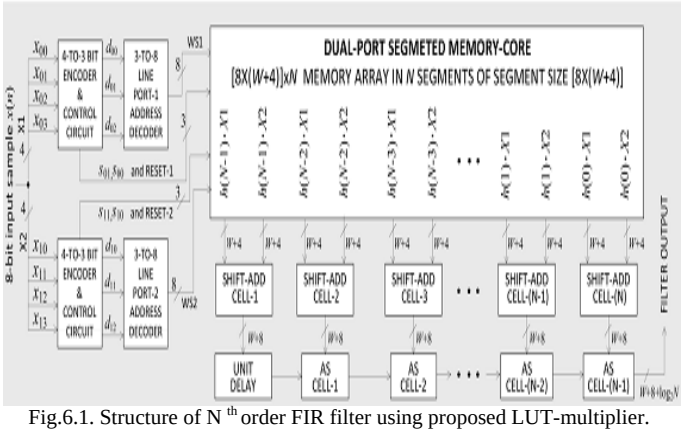

Fig.6.1. Structure of N [th] order FIR filter using proposed LUT-multiplier.

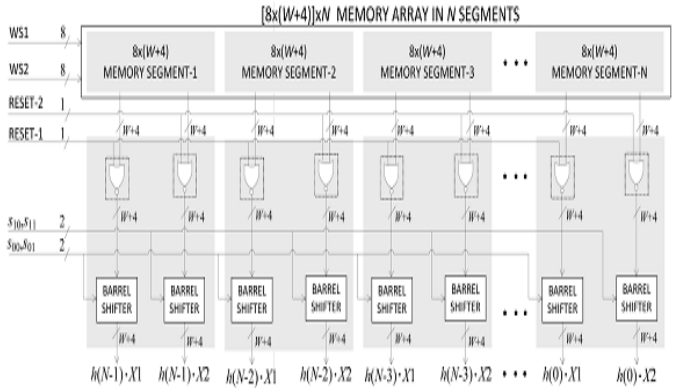

Fig.6.2.The dual-port segmented memory core for the Nth order FIR filter.

As shown in Fig.6.1 the proposed structure of FIR filter consists of a single memory-module, and an array of N shift-add (SA) cells, (N-1) AS cells and a delay register. The structure of memory module of Fig.6.1 is similar to that of Fig. 5. Like the structure of Fig.5, it consists of a pair of 4-to-3 bit encoders and control circuits and a pair of 3-to-8 line decoders to generate the necessary control signals and word select signals for the dual-port memory core.

A pair of 4-bit sub-words X1 and X2 are derived from the input sample x(n) and fed to the pair of 4-to-3 bit encoders and control circuits, which produce two sets of word-select signals (WS1 and WS2),a pair of control signals ((s_{01}, s_{00}) and (s_{11}, s_{10})) , and two reset signals. All these signals are fed to the dual-port memory-core of [8 x (W+4)] size as shown in Fig.6.1. N segments of the memory-core then produce N pairs of corresponding output, those are fed subsequently to the pairs of barrel-shifters through the 2N NOR cells. The array of N pairs of barrel-shifters thus produce N pairs of output (h(i).X1, h(i).X2) for $0 \leq i \leq N-1$. The structure and function of the NOR cells and the barrel-shifters are the same as those discussed. The structures and functions of the SA cells and AS cells are the same as those of Fig.3.1 for the structure of conventional LUT-multiplier-based FIR filter.

VII. LUT-MULTIPLIER-BASED FIR FILTER STRUCTURE BY TRANSPOSED FORM REALIZATION

We find that instead of direct-form realization, transposed form realization of FIR filter is more efficient for the LUT-multiplier-based implementation. In the transposed form, a single segmented-memory core could be used instead of separate memory modules for individual multiplications in order to avoid the use of individual decoders for each of those separate modules.

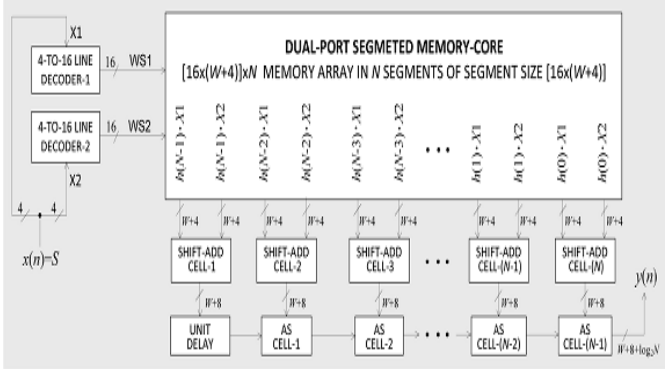

Fig.7. LUT-multiplier-based structures of an N-tap FIR filter by transposed form realization using segmented memory-core.

Since the same pair of address words X1 and X2 are used by all the LUT-multipliers in Fig.3.1, only one memory module with segments could be used instead of independent memory modules.

A conventional LUT-multiplier-based structure of an N-tap FIR filter using segmented memory-core is shown in Fig.7. It consists of dual-port segmented memory-core of size [16 x (W+4)] x N, which consists of N segments of size [16 x (W+4)]. The structure of Fig.7, involves only one pair of 4-to-16 lines decoders to receive an 8-bit input sample in each cycle, and to provide a pair of 16-bit word select signals WS1 and WS2 to the segmented memory core. The latency and throughput per cycle of this structure are the same as that of fig 6.1.

VIII. RESULTS

We have implemented the FIR filter using proposed LUT-multiplier and LUT-multiplier based transposed form FIR filter both of order four using Xilinx tool and we have presented the output for each order upto four in the simulated results.

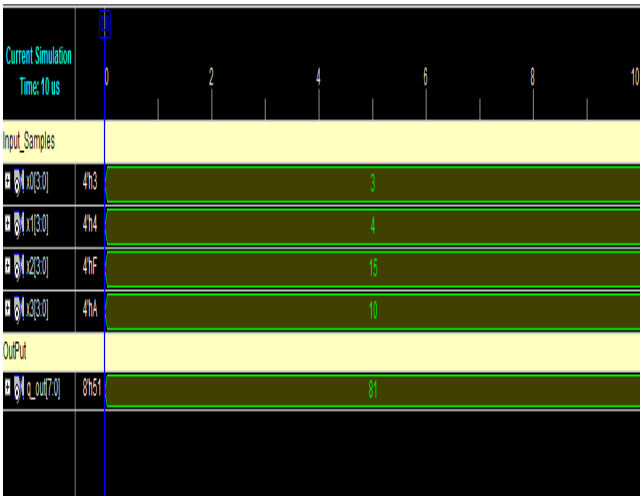

Fig.8.Simulation results of Proposed Four bit LUT multiplier

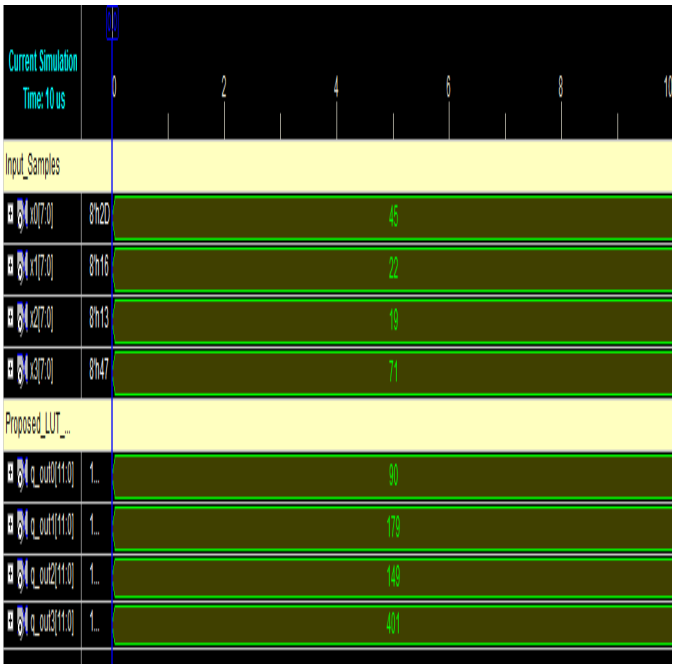

Fig.9.Simulation results of FIR filter using Proposed LUT multiplier

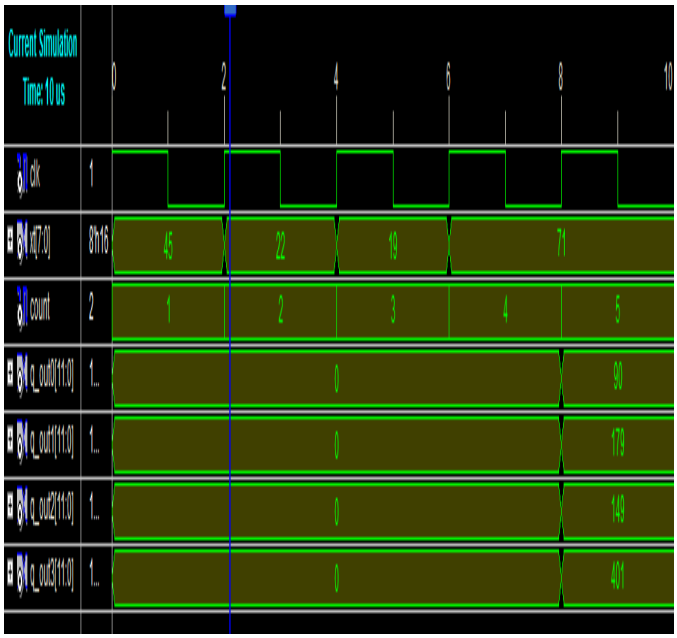

Fig.10.Simulation results of transposed form FIR filter using LUT multiplier

IX. Conclusion

The proposed LUT-multiplier-based design of FIR filter is more efficient than the previous DA and Conventional LUT based design of FIR filter in terms of area complexity for a given throughput and lower latency of implementation. Finally it is proved to be a low-complexity dedicated VLSI system for filters. Therefore LUT multipliers could be used high speed hardware implementation of digital filters and also for memory-based implementation of cyclic and linear convolutions, sinusoidal transforms, and inner-product Computation. The performance of memory based structures, with different adder and memory implementations could be studied in future for different DSP applications.

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. Upper Saddle River, NJ: Prentice- Hall, 1996.

[2] D. Xu and J. Chiu, "Design of a high-order FIR digital filtering and variable gain ranging seismic data acquisition system," in *Proc. IEEE Southeastcon'93*, Apr. 1993, p. 6.

[3] K. K. Parhi, *VLSI Digital Signal Procesing Systems: Design and Implementation*. New York: Wiley, 1999.

[4] H. H. Dam, A. Cantoni, K. L. Teo, and S. Nordholm, "FIR variable digital filter with signed power-of-two coefficients," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1348–1357, Jun. 2007.

[5] R. Mahesh and A. P. Vinod, "A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters," *IEEE Trans. Computer-Aided Ded. Integr. Circuits Syst.*, vol. 27, no. 2, pp. 217–229, Feb. 2008.

[6] *International Technology Roadmap for Semiconductors*, [Online]. Available: http://public.itrs.net/

[7] K. Itoh, S. Kimura, and T. Sakata, "VLSI memory technology: Current status and future trends," in *Proc. 25th Eur. Solid-State Circuits Conference, {ESSCIRC'99}*, Sep. 1999, pp. 3–10.

[8] T. Furuyama, "Trends and challenges of large scale embedded memories," in *Proc. IEEE Conf. Custom Integrated Circuits*, Oct. 2004, pp. 449–456.

[9] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. Mckenzie, "Computational RAM: Implementing processors in memory," *IEEE Trans. Design Test Comput.*, vol. 16, no. 1, pp. 32–41, Jan. 1999.

[10] H.-R. Lee, C.-W.Jen and C.-M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Trans. Consum. Electron.*, vol. 39, no. 3, pp. 619–629, Aug. 1993.

[11] S. A. White, "Applications of the distributed arithmetic to digital signal processing: Atutorial review," *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 5–19, Jul. 1989.

[12] M. Mehendale, S. D. Sherlekar, and G.Venkatesh, "Area-delay tradeoff in distributed arithmetic based implementation of FIR filters," in *Proc. 10th Int. Conf. VLSI Design*, Jan. 1997, pp. 124–129.

[13] H.-C. Chen, J.-I. Guo, T.-S. Chang, and C.-W. Jen, "A memory-efficient realization of cyclic convolution and its application to discrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 3, pp. 445–453, Mar. 2005.

[14] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic,"
IEEE Trans. Signal Process., vol. 56, no. 7, pp. 3009–3017, Jul. 2008.

[15] J.-I. Guo, C.-M. Liu, and C.-W. Jen, "The efficient memory-based VLSI array design for DFT and DCT," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 10, pp. 723–733, Oct. 1992.

[16] P. K. Meher, "New approach to LUT implementation and accumulation for memory-based multiplication," in *Proc. 2009 IEEE Int. Symp. Circuits Syst., ISCAS'09*, May 2009, pp. 453–456.

[17] A. K. Sharma, *Advanced Semiconductor Memories: Architectures, Designs, and Applications*. Piscataway, NJ: IEEE Press, 2003.

[18] E. John, "Semiconductor memory circuits," in *Digital Design and Fabrication*, V. G. Oklobdzija, Ed. Boca Raton, FL: CRC Press, 2008.

[19] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proc. Int. Symp. Integr. Circuits (ISIC'09)*, Dec. 2009.