Efficient Method for Look-Up-Table Design in Memory Based Fir Filters

Md.Zameeruddin
M.Tech, DECS,
Dept. of ECE,
Vardhaman College of Engineering,
Hyderabad, INDIA

Sangeetha Singh
Associate Professor,
Dept. of ECE,
Vardhaman College of Engineering,
Hyderabad, INDIA

ABSTRACT
Distributed arithmetic (DA)-based computation is well known for efficient memory-based implementation of Finite impulse response (FIR) filter where the filter outputs are computed as inner-product of input-sample vectors and filter-coefficient vector. In this paper, we show that the LUT multiplier based approach in which the memory elements store all the possible values of product of filter co-efficient will be the efficient in terms of area with the same throughput in comparison of DA. We present two new approaches to LUT-based multiplication, which could be used to reduce the memory size to half of the conventional LUT-based multiplication. The proposed method in this paper have half memory required than the existing DA method .The DA and the proposed LUT method are simulated and synthesized using the Xilinx tool and the memory required by the proposed LUT is nearly 50% lesser than the DA.

Keywords
Distributed Arithmetic (DA), FIR filter, Look-Up-Table.

1. INTRODUCTION
Filters are widely used in many applications of signal processing, the FIR digital filters are advantageous for signal processing and image processing applications[1] in the present criteria .The transition between a pass band and adjacent stop band is determined by the order of the filter .If the filter order is higher ,then there is sharper transition between pass-band and adjacent stop-band and vice-versa for the lower order filter .Many applications in digital signal processing require higher order filters[2][3] .Some of the applications involving higher order filters are frequency channelization, channel equalization, speech processing and noise elimination. The filters used in mobile systems must be of higher tap and should consume low power with high speed. As the order of the filter increases, the complexity and time consumption increases exponentially.

Now-a-days, the semiconductor industry has tremendous growth. The semiconductor memories have become cheaper, power efficient and faster. According to the requirements in different applications the memory technology has been used widely. The memories used in different applications have different uses like high reliability for biomedical instruments, low power memories for consumer products and high speed memories for multimedia applications. These memories have to be moved to processors or processors have to be moved to memory in order to minimize the bandwidth, power dissipation and access delay. The memory elements like RAM or ROM have been used either as a complete arithmetic circuit or a part of that for various applications [5]. Memory based elements are more regular when compared with the multiply-accumulate structures and have greater potential for higher throughput and reduced latency. Since the memory access time is shorter than the multiplication time in conventional multipliers, these have less dynamic power dissipation due to less switching operations. Memory based structures are suitable for digital signal processing (DSP) algorithms, which involves multiplication with a fixed set of coefficients.

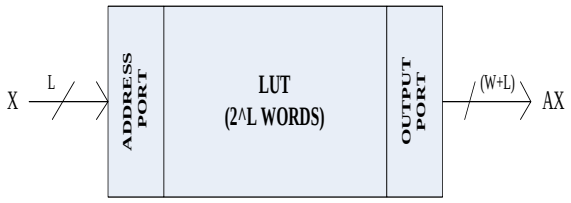

Fig 1: Conventional Memory–Based Multiplier

There are two basic types of memory based techniques. One of them is on distributed arithmetic (DA) and the other is on computation of multiplication by look-up-tables [9].The distributed arithmetic (DA) consists of inner product computation [6]-[9].In this approach, an LUT is used to store all possible values of inner products of a fixed N-point bit vector and this increases as the word length of input values increases. In LUT multiplier based approach, the multiplications of input values with a fixed –coefficient are performed by an LUT consisting of all possible pre-computed product values. Various algorithms have been implemented for efficient LUT multiplier based implementation [9], but we do not find any further way to improve the efficiency. In this paper, we aim at presenting the new approach for designing LUT multiplier based implementation where the memory size is reduced to half of the conventional approach.

The Conventional memory based multiplier is shown in Fig.1. It consists of Address port, Output port, and LUT of 2^L words. The input is X with L-bits and the output is $(W+L)$ bits. The principle of memory-based multiplication is shown in Fig 1.Let A be a fixed coefficient and X be an input word to be multiplied with *A*. If *X* is an unsigned binary number of word-length *L*, there can be 2^L possible values of *X*. Similarly, there can be possible values of product *C=A.X*. Therefore, for Conventional implementation of memory-based multiplication, a memory unit of 2^L words is to be required, which can 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 $0 < X_i < 2^L - 1$, is stored at the memory location whose address is same 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.

The even multiples 2*A*, 4*A* and 8*A* are derived by left-shift operations of *A*. Similarly, 6*A* and 12*A* are derived by left-shifting 3*A*, while 10*A* and 14*A* are derived by left-shifting 5*A* and 7*A*, 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, only (2^L/2) odd multiple values need to be stored in the memory-core of the LUT, whereas, 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 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 *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.

The 2^L possible values of *X* corresponds to 2^L possible values of *C=A.X*. The (2^L /2) words corresponding to the odd multiples of *A* may only be stored in the LUT [9].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 from 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.

Table 1: LUT words and product values for input word length L=4

Input $x_3x_2x_1x_0$	*Address* $d_2d_1d_0$	*Word symbol*	*Stored value*	*Product value*	*# of shifts*	*Control S1 S0*
0 0 0 1	0 0 0	P_0	A	A	0	0 0
0 0 1 0				2^1 x A	1	0 1
0 1 0 0				2^2 x A	2	1 0
1 0 0 0				2^3 x A	3	1 1
0 0 1 1	0 0 1	P_1	3A	2^0 x 3A	0	0 0
0 1 1 0				2^1 x 3A	1	0 1
1 1 0 0				2^2 x 3 A	2	1 0
0 1 0 1	0 1 0	P_2	5A	2^0 x 5A	0	0 0
1 0 1 0				2^1 x 5A	1	0 1
0 1 1 1	0 1 1	P_3	7A	2^0 x 7A	0	0 0
1 1 1 0				2^1 x 7A	1	0 1
1 0 0 1	1 0 0	P_4	9A	9A	0	0 0
1 0 1 1	1 0 1	P_5	11A	11A	0	0 0
1 1 0 1	1 1 0	P_6	13A	13A	0	0 0
1 1 1 1	1 1 1	P_7	15A	15A	0	0 0

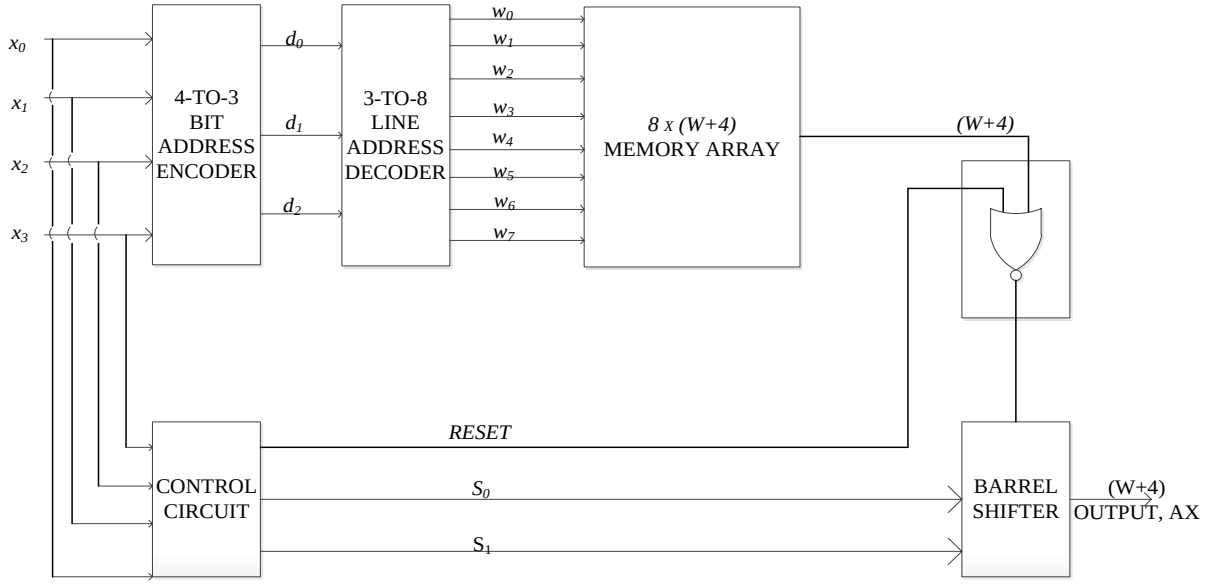

Fig 2: Proposed LUT design for multiplication of W-bit fixed coefficient

2. THE PROPOSED LUT DESIGN APPROACH FOR MEMORY BASED MULTIPLICATION

The proposed LUT design is shown in the following Fig 2.Each block in the Fig 2 is again shown in detail the internal circuit in the Fig 3 to Fig 6.

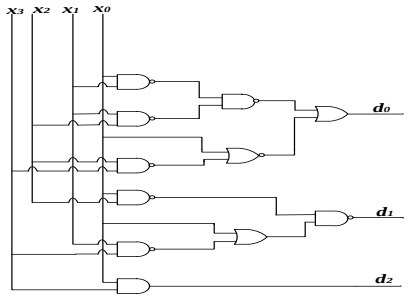

Fig 3: 4-to-3 bits input encoder

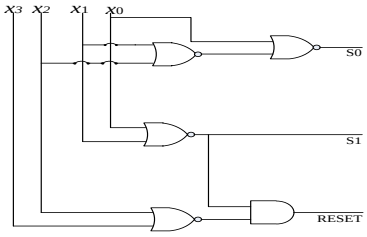

Fig 4: control circuit

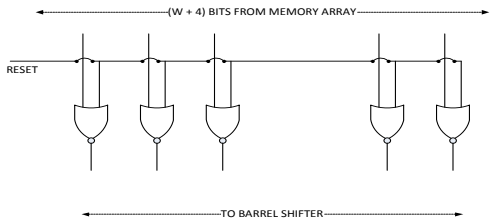

Fig 5: Structure of NOR cell

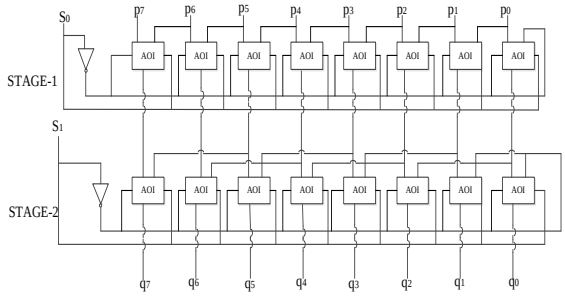

Fig 6: Two-stage logarithmic barrel-shifter for W=4

The proposed LUT based multiplier for input word-size L=4 is shown in Fig 2.It consists of 4-to-3 bit address encoder, 3-to-8 line address decoder, a memory array of eight words of (W+4) bit-width, NOR cell, control circuit and a barrel shifter. The 4-to-3 bit input encoder is shown in Fig 3. It receives 4 bit input word $(x_3x_2x_1x_0)$ and maps that into three bit address word $(d_2d_1d_0)$, according to the logic relations given below.

$$d_0 = \overline{\overline{(x_0.x_1)}.\overline{(x_1.x_2)}.(x_0 + \overline{(x_2.x_3)})} \qquad (1a)$$

$$d_1 = \overline{\overline{(x_0.x_2)}.(x_0 + \overline{(x_1.x_3)})} \qquad (1b)$$

$$d_2 = x_0.x_3 \qquad (1c)$$

These three bit address inputs are given to a decoder and it generates 8 word select signals to select the referenced-word from the memory array. The output of the memory array is either *AX* or its sub multiple in bit-inverted form depending on the value of *X*. From table I, we find that the LUT output is to be shifted to one location left when the input operand X is one of the values {(0010),(0110),(1010),(1110)}.Two left shifts are required if *X* is either (0100) or (1100).Only when input word *X*=(1000), three shifts are required. Since the maximum number of shifts required on the stored–word is three, a two-stage logarithmic barrel-shifter is adequate to perform the necessary left-shift operations. The number of shifts required to be performed on output of LUT depends on the control bits s_0 and s_1 for different values of *X* are shown in Table I. The control circuit generates the control bits by

$$s_0 = \overline{x_0 + \overline{(x_1 + x_2)}} \qquad (2a)$$

$$s_1 = \overline{(x_0 + x_1)} \qquad (2b)$$

Depending on the control bits the number of shifts is decided and implemented by the barrel shifter. A logarithmic barrel shifter of *W*=*L*=4 is shown in the Fig 6. 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 $(s_0, \overline{s_0})$ and $(s_1, \overline{s_1})$ are fed to AOI gates of stage-1 and stage-2 of barrel shifter. Since each stage of the AOI gates perform inverted multiplexing, outputs with desired number of shifts are produces in un-inverted form.

The input *X*= (0000) corresponds to multiplication by *X*=0 which results in product value *A.X*=0.So, the output of the LUT is to be reset when the input operand word *X*= (0000). The reset function is not implemented by a NOR-cell consisting of (W+ 4) NOR gates as shown in Fig 6. The inputs for the NOR gates are the RESET bit and (*W*+4) bits of LUT output in parallel. When *X*= (0000), the control bits generates active-high RESET according to the logical expression:

$$RESET = \overline{(x_0 + x_1)}.\overline{(x_2 + x_3)} \qquad (3)$$

When *RESET*=1, the outputs of all NOR gates become 0, so that the barrel shifter is fed with (*W*+4) number of zeros. When *RESET*=0, the outputs of all NOR gates become complement of the LUT output bits. The *RESET* function can be implanted by an array of 2 input AND gates, but the implementation of reset by NOR-cell is preferable since the NOR gates have simpler CMOS implementation compared with AND gates. Moreover, instead of using a separate NOR-cell, the NOR gates could be integrated with memory array if the LUT is implemented by ROM [9] [10]

2.1 Proposed 8-bit LUT Multiplier

The proposed 8-bit LUT multiplier is same as 4-bit LUT multiplier, but the difference is the usage of dual port memory array. Instead of using dual port memory array, we can use two single port memory arrays, but the dual port memory array is more efficient. The proposed 8 bit LUT multiplier is shown in following Fig 7.

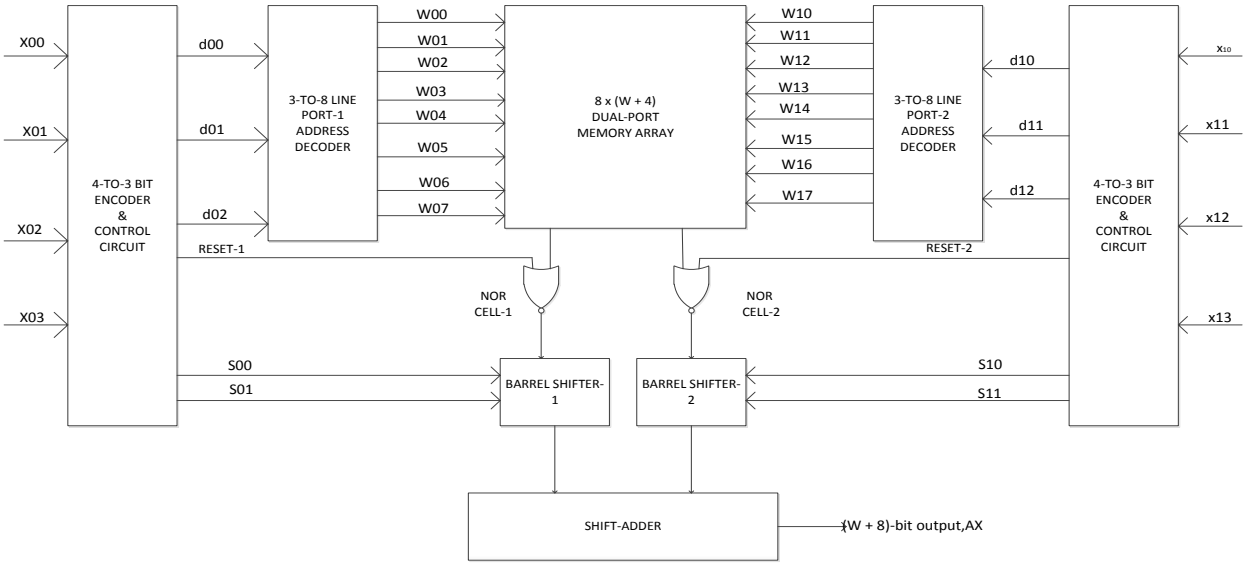

Fig 7: Memory based multiplier using dual port memory array.

The multiplication of 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 and pair of encoders, decoders, NOR cells and barrel shifter as shown in Fig 7.The shift-adder performs left shift operation of the output of barrel shifter corresponding to more significant half of input by four bit-locations, and adds that to the output of the other barrel-shifter.

3. MEMORY-BASED FIR FILTERS USING DIFFERENT METHODS.

In this section ,we are going to show the three different methods of memory-based FIR filters .In each method, different approach have been taken.

3.1 Memory based FIR filters using conventional LUT

The structure of N-tap FIR filters for input word length *L*=8 are shown in Fig 8. It consists of N memory units for conventional based multiplication, along with (*N*-1) add-subtract (*AS*) cells and a delay register. During each cycle, all the 8 bits of current input sample x(n) are fed to all the LUT-multipliers in parallel as pair of 4-bit addresses *X*1 and *X*2.The structure of the LUT multiplier is shown in Fig 8. It consists of a dual port memory unit of size [16 x (*W* +4)] and a shift–add 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 adders are required. The outputs of the multipliers are given to the pipeline of AS cells in parallel. It consists of either adder or subtract or depending on the corresponding filter weight is positive or negative. The FIR filter structure of Fig .7, takes one input sample in each cycle, and produces one filter output in each cycle. The first filter output is obtained after a latency of three cycles (one cycle each for memory output, the SA cell and the last AS cell). But the first (*N*-1) outputs are not correct because they do not contain the contributions of all the filter coefficients.

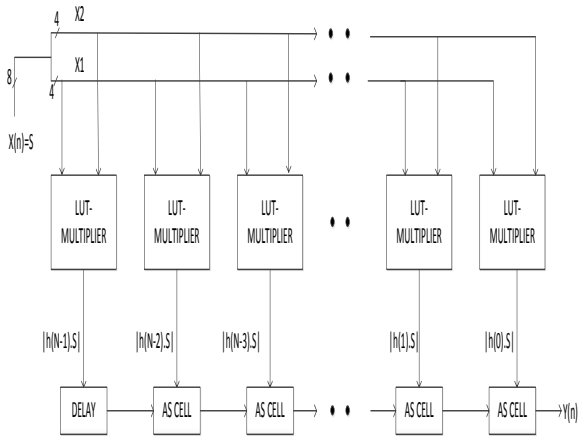

Fig 8: Conventional LUT-multiplier based structure of an N-tap FIR filter for input-width length L=8.

3.2 Memory–based FIR filter using proposed LUT design

As shown in Fig 9, 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 is same as that of 4-bit proposed LUT model consisting of 4-to-3 bit encoder, 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. The 8 bit input sample is divided as 4bit MSB and 4 bit LSB and the same process goes on as in 4 bit LUT, but here as a pair of 4 bit LUT.

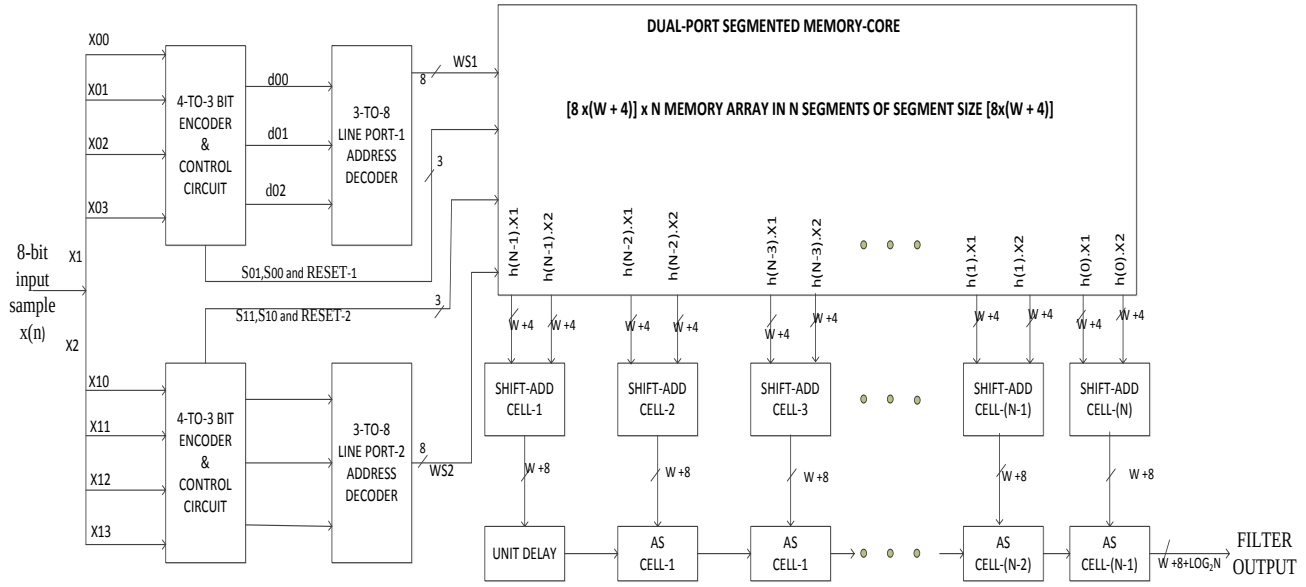

Fig 9: Structure of N the order FIR filter using proposed LUT-multiplier

The memory based structure of proposed LUT differs from conventional memory based structure in two design aspects.

1. The conventional LUT–multiplier is replaced by 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 a dual port memory.
2. Since the same pair of address words X_1 and X_2 is used by all the N LUT multipliers in Fig 9, only one memory module with N segments could be used instead of N modules. If all the multiplications are implemented by a single memory module, the hardware complexity of 2(N-1) decoder circuits can be eliminated.

3.3 DA-based implementation of FIR filter

In this section, we present the existing method of computation in FIR filters which is DA based implementation of FIR filter that has the same throughput rate as that of the LUT-multiplier based structures. Finally we found that the DA-based FIR filter structure results in minimum area and minimum area-delay product for address length 4.In Fig.10,we have shown a modified form of the 2-D structure of FIR filter presented in[8] is replaced by pipelined adder-tree and pipelined-shift-add-tree to reduce the number of latches and latency. In each cycle, one 8-bit input sample is fed to the word-serial bit-parallel converter, out of which a pair of consecutive bits are transferred to each of its four DA-based computing sections. The structure of each DA-based section is shown in Fig.10.2. The Figure consists of a pair of serial-in parallel-out bit-level shift-registers (SIPOSRs), (N/4) memory modules of size [16 x (W + 2)], (N/4) shift-add (SA) cells and a pipelined shift-adder-tree.

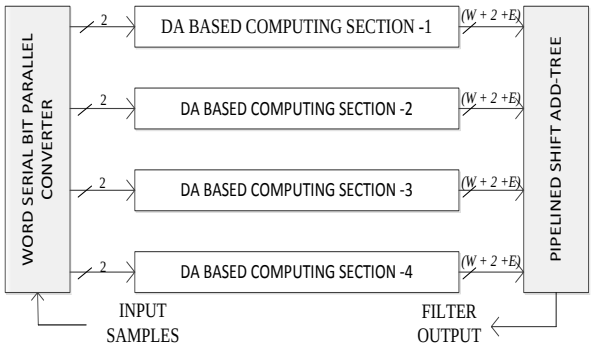

Fig 10.1: DA-based FIR filter

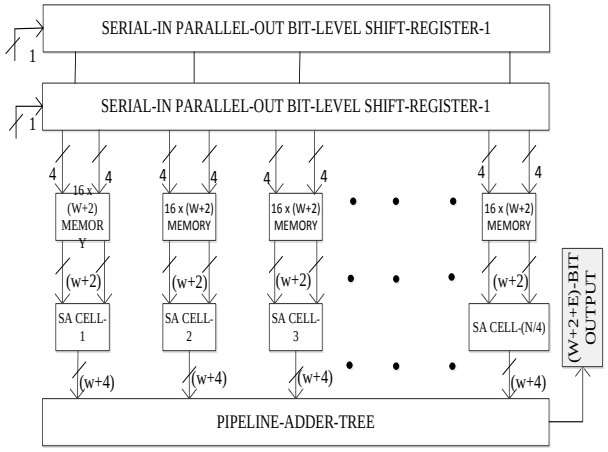

Fig 10.2: Structure of each section of filter E=log2 N

Fig 10: DA-based structure for FIR filters

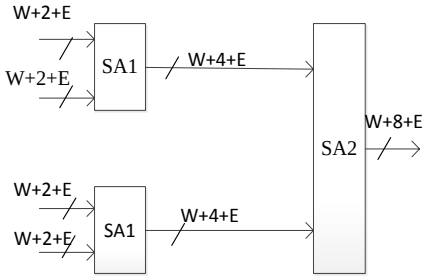

Fig 11: Pipelined shift-add-tree E=log2 N

The memory module, in each cycle, is fed with a pair of 4-bit words at the pair of address-ports. The left address-port receives 4-bit words from Serial-in parallel-out shift register-1(SIPOSR-1), whereas the right address-port receives 4 bits from the serial-in parallel-out shift register-2(SIPOSR-2).The bits at the right address port are the next significant bits corresponding to the bits available at the left address-port. According to the pair of 4-bit addresses a pair of $(W + 2)$ bit words are read-out from each memory module and fed to the *SA* cell. The *SA* cell shifts the right-input by one position to left and adds that with the left-input to produce a $(W + 4)$-bit output. The outputs of the SA cells are added by pipelined shift-add-tree consisting of three adders in two pipelined stages (shown in Fig.11). The pair of shift-adders(*SA* 1) in stage-1 shift their lower input to left by two-bit positions and add with their upper input, while the shift-adder(*SA*2) in stage-2 shifts the lower input by four-bit positions and adds that to the upper input to produce a $(W + 8 + \log_2 N)$-bit output. Therefore, the structure consists of N cycles to fill the serial-in parallel-out shift registers, one cycle for memory access and the one cycle for producing the output of the shift-add cell, $(\log_2 N - 2)$ cycles in the pipelined-adder-tree and two cycles at pipelined–shift-adder-tree. The latency for this structure is $(N + \log_2 N + 2)$ cycles, and it has the same throughput of one output per cycle same as that of the LUT-multiplier-based structures. When the input word-length is multiple of 8, such as $L=8k$ (k is integer of any value). The DA-based filter could also be implemented by k parallel sections where each section is an 8-bit filter identical to one of structures in Fig.10. The outputs of all the 8-bit filter sections are shift-added in a pipeline shift-add-tree to derive the filter outputs. The structure for $L=8k$ would have the same throughput of one output per cycle with a latency of $(N + \log_2 N + \log_2 k + 2)$ cycles.

4. RESULTS

The simulation results of the existing method, conventional LUT and proposed LUT are shown in the following Fig.12, Fig.13 and Fig.14 respectively. The synthesis reports of both conventional LUT and proposed LUT with 8 bit inputs are taken as reference and shown in the Fig.15, Fig.16 and Fig.17 respectively. On comparing both the methods, we can see the usage of the memories by individual blocks and the memory occupied by the proposed LUT is found to be low in comparison of conventional LUT. The synthesis report clearly determines the size occupied by the individual blocks and their area percentage. The DA method is taken as reference and compared with the Conventional LUT method and proposed LUT method using synthesis report. The simulation and synthesis are done in Xilinx software. In comparison, the Conventional LUT occupies 58% of total available resources, i.e. the size is reduced 42% of size compared to DA. Similarly, the proposed LUT occupies 50%, i.e. the size is reduced to 50% when compared to DA. By considering all factors, the proposed LUT method saves nearly 20% of memory than to DA method.

Fig 32: Simulation result of Distributed arithmetic

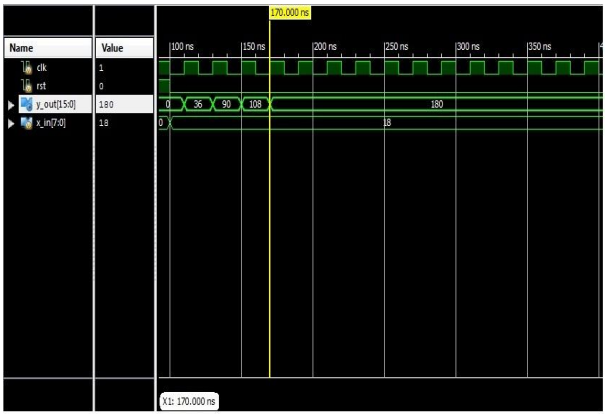

Fig 43: Simulation result of Conventional LUT

Fig 54: Simulation result of Proposed LUT design

Device Utilization summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	244	960	25%
Number of 4 input LUTs	387	1920	20%
Number of bonded IOBs	44	66	66%

Fig 65: Synthesis report of Distributed Arithmetic

Device Utilization summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	69	960	7%
Number of slice Flip Flops	48	1920	2%
Number of 4 input LUTs	127	1920	6%
Number of bonded IOBs	26	66	39%
Number of GCLKS	1	24	4%

Fig 76: Synthesis report of Conventional LUT

Device Utilization summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	62	960	6%
Number of slice Flip Flops	44	1920	2%
Number of 4 input LUTs	112	1920	5%
Number of bonded IOBs	22	66	33%
Number of CLKS	1	24	4%

Fig 87: Synthesis report of Proposed LUT

5. CONCLUSION

The modified LUT based multiplication is implemented to reduce the LUT size than that of the conventional LUT design. The LUT size is reduced to half by using two stage logarithmic barrel shifter and (*W*+4) number of NOR gates, where W is the word-length of the fixed multiplier coefficient. Two memory based structures having the unit throughput rate are designed for the implementation of the FIR filter. One is LUT based multiplier using conventional and the other is proposed LUT method. These two structures are found to have same cycle-periods, which depend on word-length, adders and filter order. The proposed LUT multiplier-based designs have half the memory than the conventional LUT design at the cost of ~4NW AOI gates and nearly ~2NW NOR gates. Therefore, the LUT multiplier based of FIR filter is more efficient than conventional in terms of area-complexity for a given throughput and low latency. These LUT based-multipliers can be used for memory based implementations of linear and cyclic convolutions, and sinusoidal transforms. The performance of memory based structures with different adders and memory can be studied in future

6. REFERENCES

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

[2] G.Mirchandani, R. L. Zinser Jr., and J. B. Evans, "A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals],"*IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process*,vol. 39, no. 10, pp. 681–694, Oct. 1995

[3] 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

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

[5] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R.Mckenzie, "Computational RAM: Implementing processors in memory," *IEEE Trans. Design Test Compute.*, vol. 16, no. 1, pp. 32–41,Jan. 1999.[13] 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

[6] 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

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

[8] 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

[9] 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.

[10] 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.

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

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