IUST
Iran University of Science and Technology

Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Research Paper

An Efficient LUT Design on FPGA for Memory-Based Multiplication

C. S. Vinitha* and R. K. Sharma*(C.A.)

Abstract: An efficient Lookup Table (LUT) design for memory-based multiplier is proposed. This multiplier can be preferred in DSP computation where one of the inputs, which is filter coefficient to the multiplier, is fixed. In this design, all possible product terms of input multiplicand with the fixed coefficient are stored directly in memory. In contrast to an earlier proposition Odd Multiple Storage (OMS), we have proposed utilizing Even Multiple Storage (EMS) scheme for memory-based multiplication and by doing so we are able to achieve a less complex and high-speed design. Because of the very simpler control circuit used in our design, to extract the odd multiples of the product term, we are also able to achieve a significant reduction in path delay and area complexity. For validation, the proposed design of the multiplier is coded in VHDL, simulated and synthesized using Xilinx tool and then implemented in Virtex 7 XC7vx330tffg1157 FPGA. Various key performance metrics like number of slices, number of slice LUT's and maximum combinational path delay is estimated for different input word length. Also, the performance metrics are compared with the existing OMS design. It is found that the proposed EMS design occupies nearly 62% less area in terms of number of slices as compared to the OMS design and the maximum path delay is decreased by 77% for a 64-bit input. Further, the proposed multipliers are used in Transposed FIR filter and its performance is compared with the OMS multiplier based filter for various filter orders and various input lengths.

Keywords: VLSI Design, Memory-Based Architecture, Multiplier, FPGA Design, FIR Filter, Transposed Structure, Distributed Arithmetic.

1 Introduction

DIGITAL Signal Processing (DSP) is the key component of the digital revolution engulfing the world of humanity. It is seen applied in almost all automated and programmable electrical appliances. The algorithms used in these DSP systems require extensive computer-based computation mostly used in real- time situation [1]. Also, DSP systems are found in portable and miniature systems which work on limited battery power and hence, these systems should occupy less silicon area [2]. Thus, DSP system designs are extremely challenging and constrained with demands such as; less area-complexity, low power dissipation and high speed of operation. It is not possible to design systems that satisfy all the constraints of the application. Architectural transformation can be done to trade-off one constraint over another like area over speed and so on. Certain solutions can be found in [3] which help reduce the arithmetic complexities of the algorithms so that the speed, area and power complexity can be addressed.

Most of the DSP algorithms use Multiply and Accumulate (MAC) operator repeatedly [4, 5]. The conventional logic based multiplier is very complex and occupy most of the space of a DSP system. Hence in the past three decades lot of research has been carried out in decreasing the complexity of the multiplier. As per the researches, the implementation of multiplier for DSP

Iranian Journal of Electrical and Electronic Engineering, 2019.
Paper first received 27 November 2018, revised 08 March 2019, and accepted 11 March 2019.
* The authors are with the Electronics and Communication Engineering Department, Ambedkar Institute of Advanced Communication Technologies and Research (GGSIP University), Geeta Colony, Delhi, India.
E-mails: csvinitha1972@aiactr.ac.in and rksharma@aiactr.ac.in.
Corresponding Author: R. K. Sharma.

application is broadly classified into three categories: CORDIC implementation, adder-based implementation and Memory-based implementation [6-11]. Among these three, Memory-based implementation is gaining more popularity due to significant growth in VLSI memory technology. Semiconductor memory has become cheaper, faster and is available with less power consumption because of the continuous improvement in silicon scaling technology. Owing to advancement in memory technology [12] and as per application requirement better memory designs are possible nowadays and hence, efficient memory-based multipliers are also not unthinkable. Earlier, memory used to be a separate section from the processor unit, but nowadays memory is becoming part of the processor. This increases the bandwidth of operation and reduces the power consumption [13]. As per projections of International Technology Roadmap of Semiconductors (ITRS), the density of DRAM cells is increasing steadily as compared to transistors in Micro Processor Unit (MPU) [12]. Thus, cost of storing one-bit information in DRAM is far less as compared to MPU. Hence, Memory- based computing is now being considered cheaper than the conventional logic-only arithmetic circuits. In [14], a brief survey report on memory-based VLSI architectures for digital filters can be found.

In DSP algorithms mostly one input to the multiplier, that is, the coefficients are fixed in nature. This facilitates the use of memory-based structures in DSP. Numerous architectures are proposed on the application of memory-based structures in discrete sinusoidal transforms and filters [10, 11], [15-40]. There are two types of memory based computation. One is using Direct- LUT to compute the multiplication [19-23], [25-31], [40] and the other is to compute the inner-product using Distributed Arithmetic (DA) [10], [16-18], [32-39]. In Direct-LUT-based computation, all the possible product terms of the input multiplicand with the fixed coefficients are pre-computed and stored directly in the LUT and thus multiplication is done [11]. But in DA-based computation, the inner-product of the N-point vector with the N-bit vector is pre-computed and stored in LUT [10]. The size of the LUT increases with the word length of the input if the product term is directly stored in LUT, whereas if the inner product is stored, the size increases with the length of the inner-product.

In DA-based computation, offset binary coding [10, 37] and group distributed [17] technique are proposed to decrease the size of the memory. Under Direct-LUT based computation many techniques are proposed [26-31], [40]. In [26, 28] and [29], authors have proposed OMS approach, where only the odd multiple product terms are stored in memory. Thus the size of the memory is reduced by half. Further, in [27, 30] another technique namely; Anti-symmetric Product Coding (APC) has been detailed, where the size of the LUT is reduced again. Also, it requires less over-head circuits as compared to the work of [26]. In [29] authors have combined both OMS and APC technique and developed an efficient architecture which contains the advantages of both of the above techniques. In [28] author has slightly modified the design used in [29] and used the proposed multiplier in FIR filter and made a comparison with the conventional and DA based memory multiplier. Also we find many papers on Finite Impulse Response (FIR) filters exploiting features of these memory- based multipliers are published [28], [32-36]. DA-based multipliers are used in Adaptive FIR filter to improve its computational efficiency and decrease its implementation complexity [42-48].

Because of OMS-based multiplier already existing, we were curios to try EMS- based multiplier design, not explored and dealt with up until now. This communication is nothing but an outcome of the same curiosity. We could convincingly implement a memory-based multiplier using EMS-design with a very less complex logic circuit. Also we could achieve a design with a very less data path delay as compared to the previous designs. Even though the work is incremental to the work in [28], it is also novel and we have proved here that our proposed EMS multiplier is area-efficient and high speed design in comparison to OMS design. The EMS multiplier [41] is used in systolic architecture for FIR filters to reduce the latency and area of the filter. We have tried in this communication to modify the already proposed OMS design with a different external logic circuit. The modification of the logic and control circuit in OMS design also resulted in an efficient multiplier with reduction in area complexity and data path delay. Hence using our proposed logic and control circuit, we are able to achieve an efficient EMS and OMS design for a memory-based multiplier. For validation of our above stated propositions, we have done coding of the proposed multipliers in VHDL; simulated and synthesized them using Xilinx tool and then finally, implemented them in Virtex 7 XC7vx330tffg1157 FPGA. Various key performance metrics like number of slices, number of slice LUT's and maximum combinational path delay has been estimated for different input word length.

The remainder of this article is organized as follows: In Section 2, the conventional Memory-based multiplier is discussed. In Section 3, the proposed EMS-based LUT multiplier is explained in detail for 4-bit and 8-bit input. The Synthesis results pertaining to FPGA implementation and its comparison with OMS technique [28] are also presented and discussed in Section 3. In Section 4, the memory–based transposed FIR filter structure using proposed LUT multiplier has been detailed out and compared with OMS design based filter. Finally, in Section 5 the work has been concluded.

2 Conventional Memory-based Multiplier

In a conventional Memory-based multiplier,

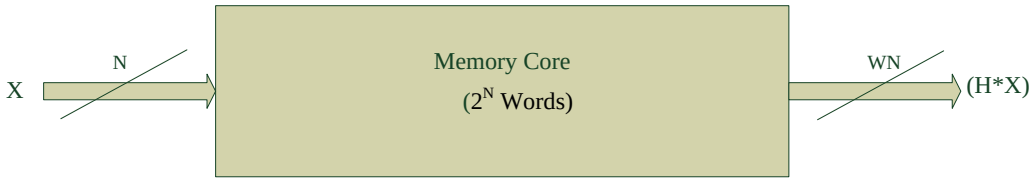

Fig. 1 Conventional memory-based multiplier.

Table 1 Conventional LUT multiplier table.

Address Bits				Content of the memory
X_3	X_2	X_1	X_0	
0	0	0	0	0
0	0	0	1	H
0	0	1	0	2H
0	0	1	1	3H
0	1	0	0	4H
0	1	0	1	5H
0	1	1	0	6H
0	1	1	1	7H
1	0	0	0	8H
1	0	0	1	9H
1	0	1	0	10H
1	0	1	1	11H
1	1	0	0	12H
1	1	0	1	13H
1	1	1	0	14H
1	1	1	1	15H

applicable for fixed coefficient, all the possible product terms are stored in different locations of the LUT. The product terms for all the possible combination of input multiplicand with the fixed coefficient are pre-calculated and stored in different locations in the memory. Let us assume that *W* is the word length of the fixed coefficient *H* and *N* is the word length of the input multiplicand *X*. The number of locations occupied by the memory depends on the size of the input *X*. For an *N*-bit input *X*, 2^N locations are occupied by the product terms. The contents of the memory for an input word length of 4-bit and for a fixed coefficient *H* is given in Table 1.

As shown in Fig. 1 the product terms ($H*X_i$) where *i* varies from zero to (*N*-1) are stored in various locations of the memory and fetched using input *X* as the address. In the next section we explain two new propositions to decrease the size of the memory and to improve the complexity of the multiplier.

3 Proposed LUT-based Multiplier for DSP Computation

Here two designs are proposed to reduce the complexity of the LUT-based multiplier. The proposed designs are named as EMS-LUT multiplier where only the even product terms are stored in memory and Modified OMS (MOMS) multiplier where only the odd product terms are stored in memory. Both of these techniques reduce the memory size by half. The concept of storing the product terms in memory is same as that proposed in [26] and [28]. The changes in our proposed design are the use of a new logic and the control circuit to derive their corresponding complement product terms. However, as will be evident from the following details, the complexity of logic and the external circuit used in our proposed design is simpler in comparison to [28].

3.1 Proposed EMS-LUT based Multiplier for 4-bit and 8-bit Input.

In this multiplier, even multiplies of the product terms are stored in memory. An external combinational logic circuit is used to derive the odd multiplies of the product term using the even product term. The working of the proposed design is explained as follows:

As discussed in Section 2, all the possible product terms are stored in memory for a conventional memory-based multiplier. For an *N*-bit input, there will be 2^N product terms. Among the $\mathbf{2^N}$ product terms, $((2^N/2) - 1)$ terms correspond to the even multiples of *H*. In this multiplier we store only these even product terms and the term 0. Hence only $(2^N/2)$ memory locations are required to store the product terms thus reducing the size of the LUT by half.

The proposed method is explained by considering a 4-bit input size. From the contents of the conventional LUT multiplier as shown in Table 1, we can see that there are seven even product terms ($H*(2i)$) where *i* varies from 1 to 7. These seven even product terms along with 0 are stored in the memory of the proposed multiplier. The contents of the proposed LUT multiplier are shown in Table 2.

Table 2 Proposed EMS-LUT multiplier table.

Address for EMSLUT $X_3X_2X_1$	Content of the memory	Input $X_3X_2X_1X_0$	Product value
000	0	0000	0
		0001	H
001	2H	0010	2H
		0011	3H
010	4H	0100	4H
		0101	5H
011	6H	0110	6H
		0111	7H
100	8H	1000	8H
		1001	9H
101	10H	1010	10H
		1011	11H
110	12H	1100	12H
		1101	13H
111	14H	1110	14H
		1111	15H

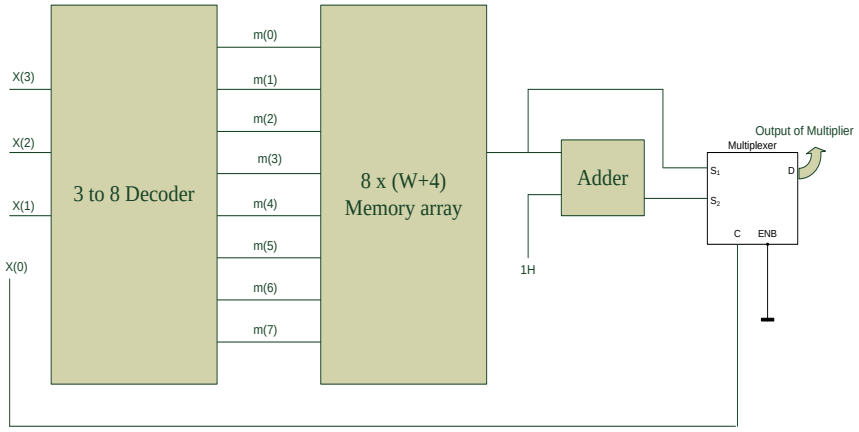

Fig. 2 Proposed EMS-LUT multiplier for 4-bit input.

For deriving the odd product terms we require the following external circuitry:

- A memory with $2^{N/2}$ memory locations with each location of width ($W+N$), where W is the width of the fixed coefficient and N is the width of the input.
- A 3-to-8 decoder to address the eight memory locations of the memory.
- An adder circuit to derive the odd product term using the even product term fetched from the memory.
- A 2 to 1 multiplexer (mux) to do selection among the even and odd product terms and pass it to the output.

The block diagram of the proposed EMS-LUT multiplier for 4-bit input is shown in Fig. 2. Initially, the pre-calculated even product terms including 0 are stored in the memory. The three Most Significant Bits (MSB) [X_3 X_2 X_1] of the input are applied as input to a 3 to 8 decoder. The 8 output lines of the decoder are used as address to fetch the product terms from the memory. For example, for the input combination [0 0 1] of the decoder, the output line m (1) of the decoder will be active and the product term (2H) is fetched out. Similarly, for the combination [0 1 0] the output line m (2) of the decoder will be active and the product term (4H) is fetched out. Likewise, it goes on and on for the last 3-bit combination [1 1 1] product term (14H) is fetched out. The output from the memory is fed directly to input line s_1 of the 2-to-1 multiplexer. The output from the memory is added with (1H) in an adder circuit and the output of the adder gives the odd multiple of the product term. The output of the adder is directly connected to the input line s_2 of the 2-to-1 multiplexer. The control line c of the multiplexer is directly connected to the Least Significant Bit (LSB) [X_0] of the input data X. Hence when the control line is 0, the mux selects the input s_1 and when the control line is 1, it selects the input s_2.Thus when the input bit X_0 equals 0, even product term is available at the output of the multiplier and when X_0 equals 1, odd product term is available at the output of the multiplier. The enable line of the mux, which is not required in this design, is grounded. For example, when the X input equals [0 0 1 0], because of the first three bit equal to [0 0 1],

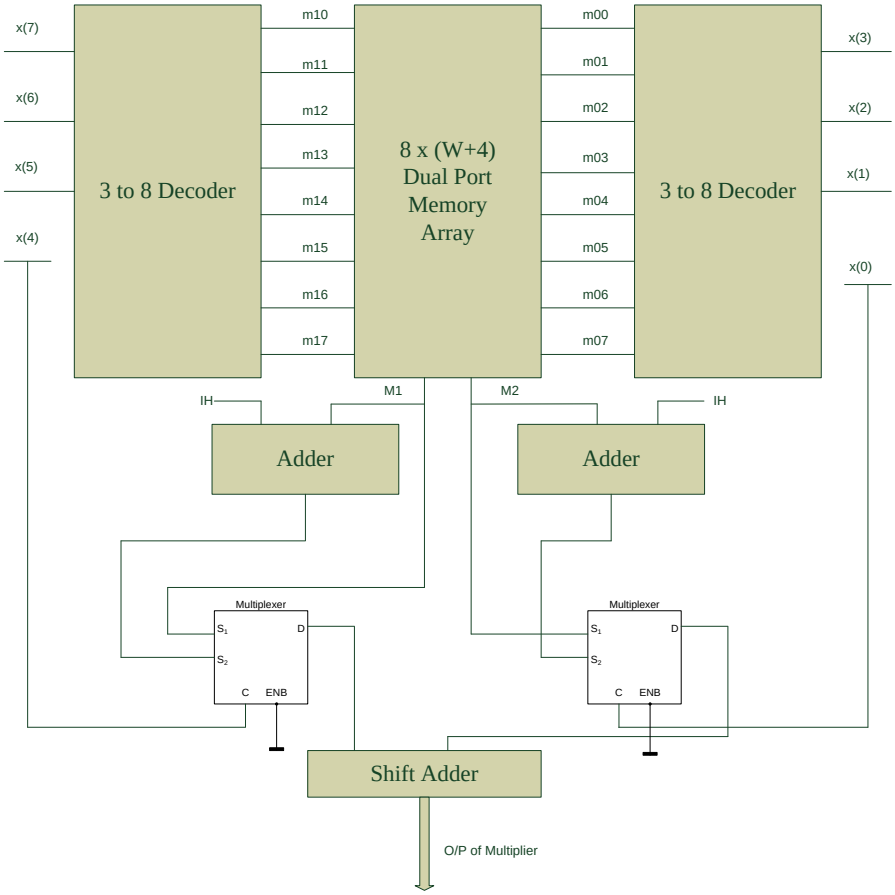

Fig. 3 Proposed EMS-LUT multiplier for 8-bit input.

the address line $m(1)$ of the memory is activated and the product term ($2H$) from the memory is selected and fed to the input line s_1 of the mux. Since, the LSB bit of the input X is 0 for this particular combination of X, the input line s_1 is selected by the mux and transferred to the output of the multiplier. The odd product term is selected as follows. For example, if the input data X is equal to [0 0 1 1], then, because the first three bits are [0 0 1], the output line m (1) of the decoder is activated which then selects the product term ($2H$) from the memory. Since the LSB of the input X is equal to 1, the input line s_2 is selected by the mux. As the input line s_2 is connected to the output of the adder, the odd product term for this case which is [$3H$] is selected by the mux and transferred at the output of the multiplier. Similarly, for the input combination [1 1 1 0] the even product term ($14H$) is transferred at the output and for the input combination [1 1 1 1] the odd product term is selected and transferred at the output. Thus, for all the possible input combinations their corresponding product terms are selected and transferred to the output of the EMS-LUT multiplier.

Using Dual Port Memory array and two section of external control circuit we can multiply 8-bit input with W-bit coefficient. Here we use two adders and two mux one each to derive the MSB (M_1) and LSB (M_2) part of the product term. Finally, a shift adder is used in which the MSB part of the product term is shifted left by four bits and then added with the LSB part of the product term. Thus the final product term is derived from the shift adder. The self-explanatory illustration of EMS-LUT Multiplier for the 8-bit input is given in Fig. 3.

3.2 Proposed Modified OMS-based Multiplier for 4-bit and 8-bit Input

Here we propose a new logic for the already existing OMS multiplier [28] design. Since it is the modification of the OMS-LUT multiplier proposed earlier, we are naming it MOMS-LUT multiplier. As compared to the proposed EMS-LUT multiplier, there is variation in the external control circuit used to derive the even product terms. The proposed design of MOMS-LUT based multiplier is explained as in Table 3.

As discussed in the previous section, let H be the fixed coefficient and X be the input multiplicand. Let N be the word length of the input X. Hence there will be 2^N possible values of input and accordingly 2^N product terms. Among the 2^N product terms, (2^N/2) terms correspond to the odd multiples of H. In this design we store only these odd product terms. Hence only (2^N/2) memory locations are required to store the product terms. Thus the size of the LUT is reduced by half.

In the following, the proposed method is explained by

Table 3 Proposed modified OMS-LUT multiplier table.

Address for OMSLUT $X_3X_2X_1$	Content of the memory	Input $X_3X_2X_1X_0$	Product value
		0000	0
0001	H	0001	H
		0010	2H
0011	3H	0011	3H
		0100	4H
0101	5H	0101	5H
		0110	6H
0111	7H	0111	7H
		1000	8H
1001	9H	1001	9H
		1010	10H
1011	11H	1011	11H
		1100	12H
1101	13H	1101	13H
		1110	14H
1111	15H	1111	15H

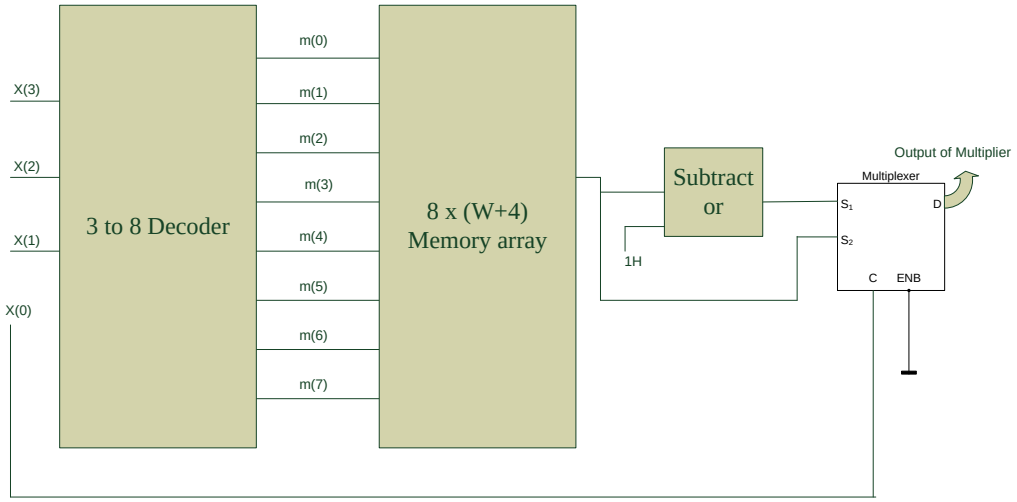

Fig. 4 Proposed modified OMS-LUT multiplier for 4-bit input.

considering a 4-bit input. From Table 1, we can see that there are eight odd product terms (H*(2i-1)) where i varies from 1 to 8. These eight odd product terms are stored in the memory of the proposed multiplier. The contents of the proposed LUT multiplier are shown in Table 3.

For deriving the even multiplies of the product term we require the following external circuitry:

- A memory with $2^{N/2}$ memory locations with each location of width (W+N), where W is the width of the fixed coefficient and N is the width of the input.
- A 3-to-8 decoder to address the eight memory locations of the memory.
- A Subtractor circuit to derive the even product term using the odd product term fetched from the memory.
- A 2 to 1 multiplexer (mux) to do selection among the even and odd product terms and pass it to the output.

The proposed multiplier for 4-bit input is shown in Fig. 4. For an input size of 4-bit, how the MOMS-LUT multiplier design works is explained in the following. Initially, the entire odd product terms are stored in the memory. The three MSB'S [X_3 X_2 X_1] of the input are applied to the 3 to 8 decoder. The 8 output lines of the decoder are used as address to fetch the product terms from the memory. For example, for the input combination [0 0 1] of the decoder, the output line m (1) will be active and the product term (3H) is fetched out. Similarly, for the combination [0 1 0] product term (5H) is fetched out. Likewise, it goes on and for the last 3-bit combination [1 1 1] for which the product term (15H) is fetched out. The output from the memory is fed directly to input line s_2 of the 2-to-1 multiplexer. The output from the memory is fed to a subtractor where *(1H*) is subtracted from the output of the memory and the output of the subtractor gives the even product term. The output of the subtractor is directly connected to the input line s_1 of the 2-to-1 multiplexer. The control line c of the multiplexer is connected to the LSB [X_0] of the input data X. Hence when the control line is 0, the mux

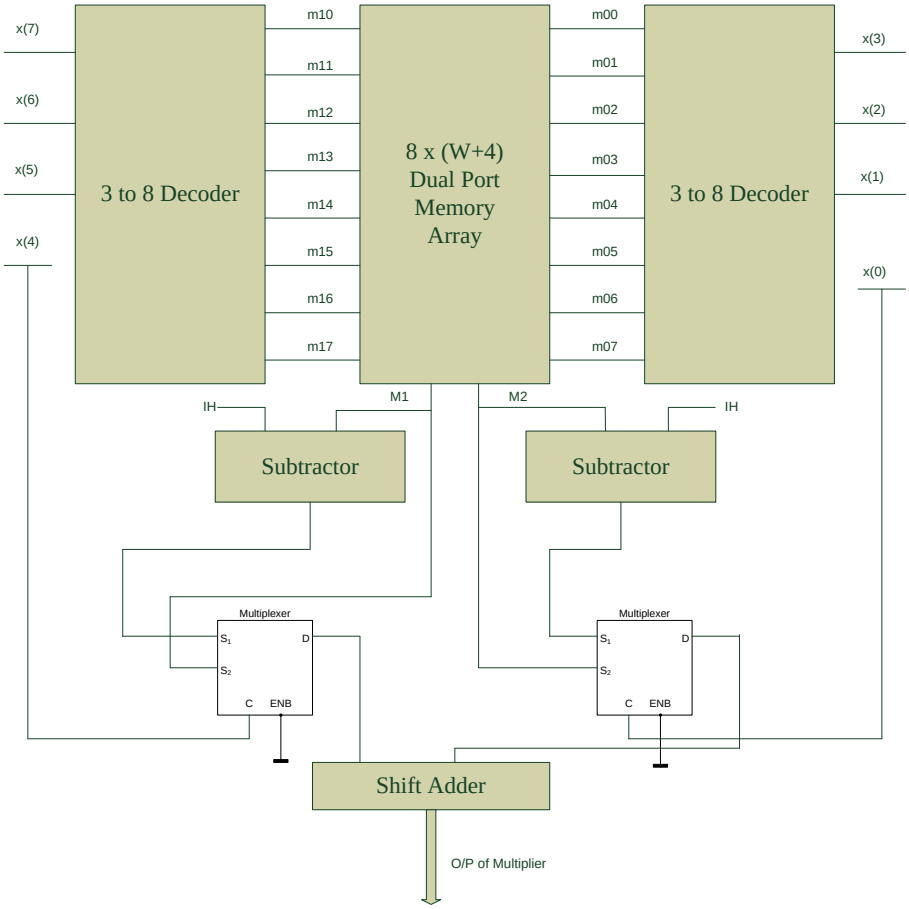

Fig. 5 Proposed modified OMS-LUT multiplier for 8-bit input.

selects the input s_1 and when the control line is 1, it selects the input s_2.Thus when the input bit X_0 equals *0*, even product term is available at the output of the multiplier and when X_0 equals 1, odd product term is available at the output of the multiplier. The enable line of the mux is grounded. For example, when the *X* input equals [0 0 1 1], because of the first three bit equal to [0 0 1], the address line *m*(1) of the memory is activated and the product term (*3H*) from the memory is selected and fed to the input line s_2 of the mux. Since, the LSB bit of the input *X* is 1 for this particular combination of *X*, the input line s_2 is selected by the mux and transferred to the output of the multiplier. The even product term is selected as follows. For example, if the input data *X* is equal to [0 0 1 0], then, because the first three bits are [0 0 1], the output line *m* (1) of the decoder is activated which then selects the product term (*3H*) from the memory. Since the LSB of the input *X* is equal to 0, the input line s_1 is selected by the mux. As the input line s_1 is connected to the output of the subtractor, the even product term for this case which is [2*H*] is transferred at the output of the multiplier. Similarly, for the input combination [1 1 1 1] the odd product term (15*H*) is transferred at the output and for the input combination [1 1 1 0] the even product term is selected and transferred at the output. Thus, for all the possible input combinations their corresponding product terms are selected and transferred to the output of the MOMS-LUT multiplier.

Using Dual Port Memory array and two section of external control circuit we can multiply 8-bit input with *L*-bit coefficient. Here we use two subtractors and two mux one each to derive the MSB (M_1) and LSB (M_2) part of the product term. Finally, a shift adder is used in which the MSB part of the product term is shifted left by four bits and then added with the LSB part of the product term. Thus the final product term is derived from the shift adder. The block diagram of MOMS-LUT Multiplier for the 8-bit input is given very clearly in Fig. 5.

3.3 LUT-based Multiplier for Higher Order Input Multiplicand

The LUT-based multiplier for the higher order inputs is synthesized using parallel realization of lower order multipliers. For example the multiplier for a 32-bit input is designed using a parallel realization of two 16-bit multipliers. Further this 16-bit multiplier is realized using two 8-bit multiplier and the 8-bit multiplier using two 4-bit multiplier. The block diagram of a 32-bit multiplier and 16-bit multiplier is given in Fig. 6.

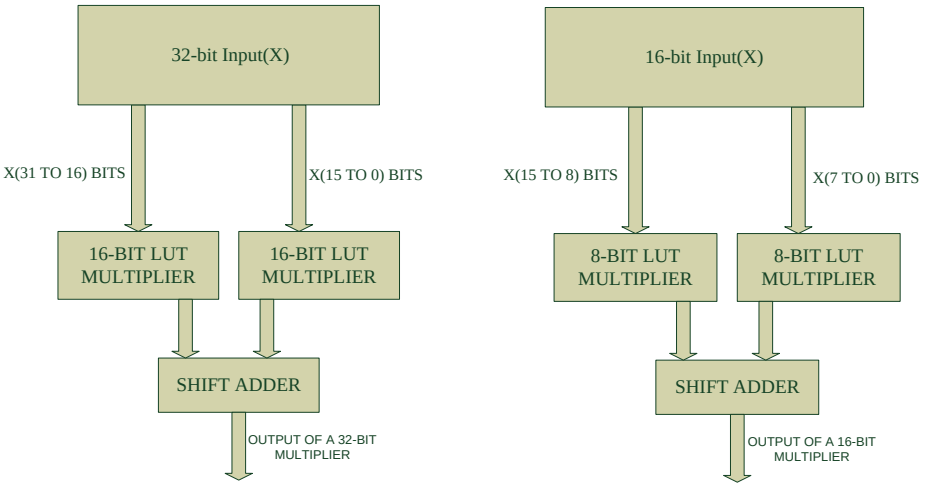

Fig. 6 Block diagram of 32-bit and 16-bit LUT-based multiplier.

Table 4 Comparison of hardware parameters of conventional and proposed multipliers for various input lengths.

Description	Length of the input							
	8-bit		16-bit		32-bit		64-bit	
	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's
Conventional memory-based design	11	24	23	48	48	96	96	192
Both MOMS & EMS design	04	09	09	18	15	36	34	72

Table 5 Comparison of hardware utilisation parameters of the multiplier for various input lengths.

Description	Length of the input							
	8-bit		16-bit		32-bit		64-bit	
	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's
Conventional memory-based design	11	24	23	48	48	96	96	192
OMS design [28]	10	30	19	59	36	119	90	240
Both MOMS & EMS design	04	09	09	18	15	36	34	72

3.4 Complexity Analysis of the Proposed LUT-based Multiplier

The conventional LUT-based multiplier for 8-bit input and for a *W*-bit coefficient requires one dual-port memory with 16 memory locations of size (*W*+4), two (4:16) decoders and one shift-adder of size (*W*+8) bit. The proposed multipliers require one dual-port memory with 8 memory locations of size (*W*+4), two (3:8) decoders, two ((*W*+8)/2) bit adder in case of EMS design or two ((*W*+8)/2) bit subtractors in case of MOMS design, two (2 to 1) mux and one shift-adder of size (*W*+8) bit. The conventional LUT-based multiplier and the proposed multipliers are coded in VHDL and synthesised using Xilinx tool and implemented in Virtex 7 XC7vx330tffg1157 FPGA. The comparison of the hardware utilised by the conventional multiplier and the proposed multiplier for various input word length is given in Table 4.

From the Table 4 we can conclude that as the word length of the input increases, the design-complexity of the proposed multipliers are consuming less hardware compared to the conventional LUT-based multiplier. For an input size of 64-bit the percentage of area saving in terms of the number of slices in the case of proposed multipliers as compared to the conventional multiplier is 62%.

3.5 Novelty and Comparison of the Proposed EMS and MOMS Multiplier with OMS Multiplier

The proposed multipliers are designed for various input lengths and simulated, synthesized and implemented using Xilinx tool. The OMS design proposed in [28] is also simulated and implemented on the same platform and the performance characteristics are compared in the following Table 5.

In OMS design proposed in [28] author has stored the odd product term in memory thereby reducing the memory size by half. But the external over-heads required for deriving the even product terms for a 4-bit input multiplicand are:

- A memory with $2^{N/2}$ memory locations with each location of width (W+N), where W is the width of the fixed coefficient and N is the width of the input.
- A 4 to 3 encoder.
- A 3 to 8 decoder.
- A barrel shifter whose size depends on the word length of the memory output.
- A control circuit to control the number of shifts in the barrel shifter.

A reset circuit to reset the output when the product term is equal to zero. But in our work we have stored the even product terms in the memory in the case of EMS design and odd product terms in the case of MOMS design to decrease the size of the memory. In the case of EMS design we have stored the even product term and the product term *zero* also. In MOMS design the product term 0 is derived from the odd product term [1H] stored in the memory. So there is no need of reset circuit to derive the output when the input is zero. Also, the odd product terms and the even product terms as per design are derived with a different logic which requires less external over-head as compared to OMS design [28], which consist of a 3-to-8 decoder, a 2-to-1 mux and an adder/ subtractor whose width depends on the word length of the memory output. In order to compare the performance and hardware utilisation of our design and the OMS-LUT design, we faithfully coded both the design in VHDL and implemented in Xilinx Virtex7 FPGA under same place-and-route conditions and using same user constraints and under same operating conditions which can be selected in the Xilinx tool. The hardware utilisation parameters like number of CLB slices, number of slice LUT's are compared for both the designs for various input word length. The comparison of the hardware utilisation parameters for different input lengths is given in Table 5. Also the area-efficiency of the proposed multiplier with the OMS design is also tabulated in Table 6 for different input size.

Similarly, the comparison of Maximum combinational path delay of both the design is given in Table.7. We are able to achieve very less path delay in our design since the input has to go through less number of logic circuits before reaching the output point. The input data passes through a decoder followed by memory, adder and a multiplexer. But in the design of [28], the data passes through encoder followed by decoder, memory, barrel shifter which has to wait for a control circuit to control the shifting of the barrel shifter, and a reset circuit to reset the output. Since the higher order input length multipliers are realised in parallel, the path delay remains same for all input lengths.

The complexity of the multipliers is compared at logic gate level also. All the three design require same capacity multiplier as we are storing only half of the

Table 6 Area-efficiency of the proposed multiplier.

Description	Length of the input			
	8-bit	16-bit	32-bit	64-bit
	No. of slices	No. of slices	No. of slices	No. of slices
OMS design [28]	10	19	36	90
Both MOMS & EMS design	04	09	15	34
Saving of area [%]	60	53	58	62

Table 7 Comparison of path delay for both the designs.

Description	Both MOMS & EMS design [ns]	OMS [ns] [28]
Maximum combinational path delay	0.339	1.495

Table 8 Hardware complexities of proposed EMS-design, proposed modified OMS-design, conventional design, OMS-design [28] based multiplier [word-length of the fixed coefficient-W, word length of the input-8]

Designs	Conventional design	OMS design [28]	This work	
			Proposed EMS design	Proposed modified OMS design
Memory	16[W+4] bit memory	8[W+4] bit memory	8[W+4] bit memory	8[W+4] bit memory
Decoder	2[4:16] decoder	2[3:8] decoder	2[3:8] decoder	2[3:8] decoder
Encoder	Not required	2[4 to 3] encoder	Not required	Not required
Adder/ Subtractor	[W+8] bit adder	[W+8] bit adder	[W+8] bit adder 2([W+8]/2) bit adder	W+8] bit adder 2([W+8]/2) bit adder
Multiplexer	Not required	Not required	2 [2 to 1] mux	2 [2 to 1] mux
NOR gates	Not required	2([W+4]-2 bit) NOR gates	Not required	Not required
AOI gates	Not required	2([W+4]-2 bit) AOI gates	Not required	Not required
Control ckts	Not required	2[2 input] NOR gates 2[2 input] OR gates	Not required	Not required

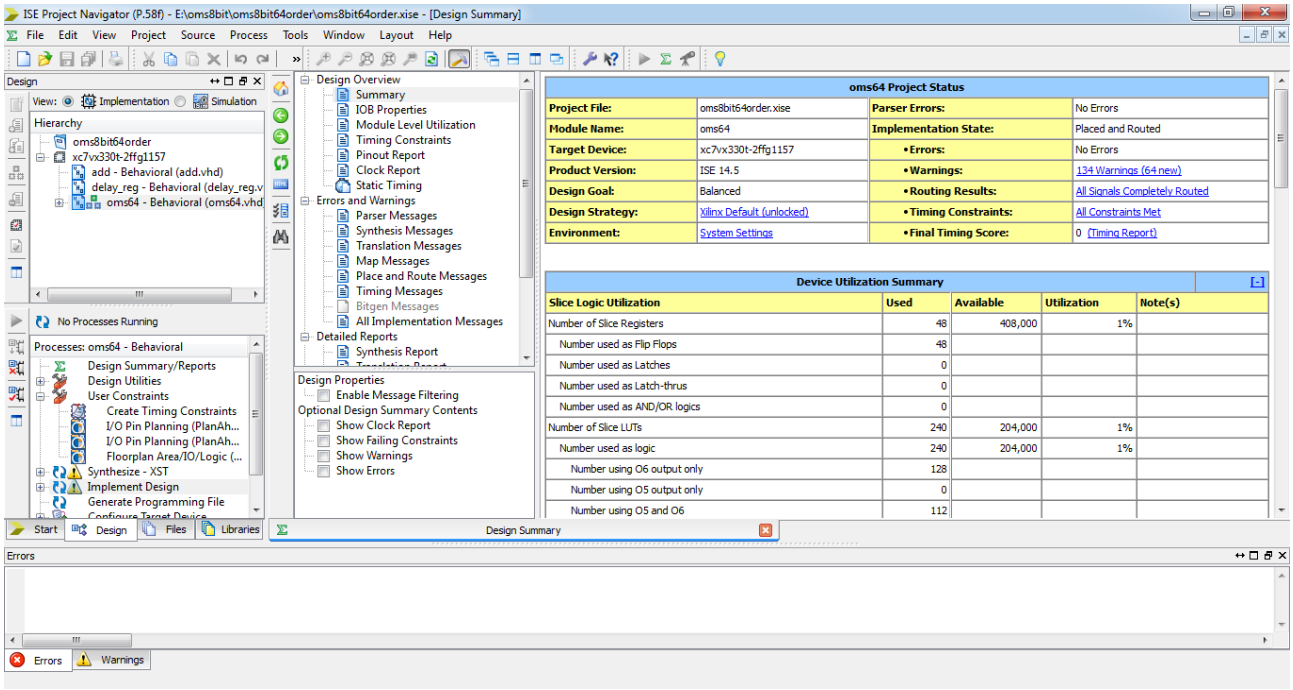

Fig. 7 Device utilization summary of OMS multiplier with 64-bit input.

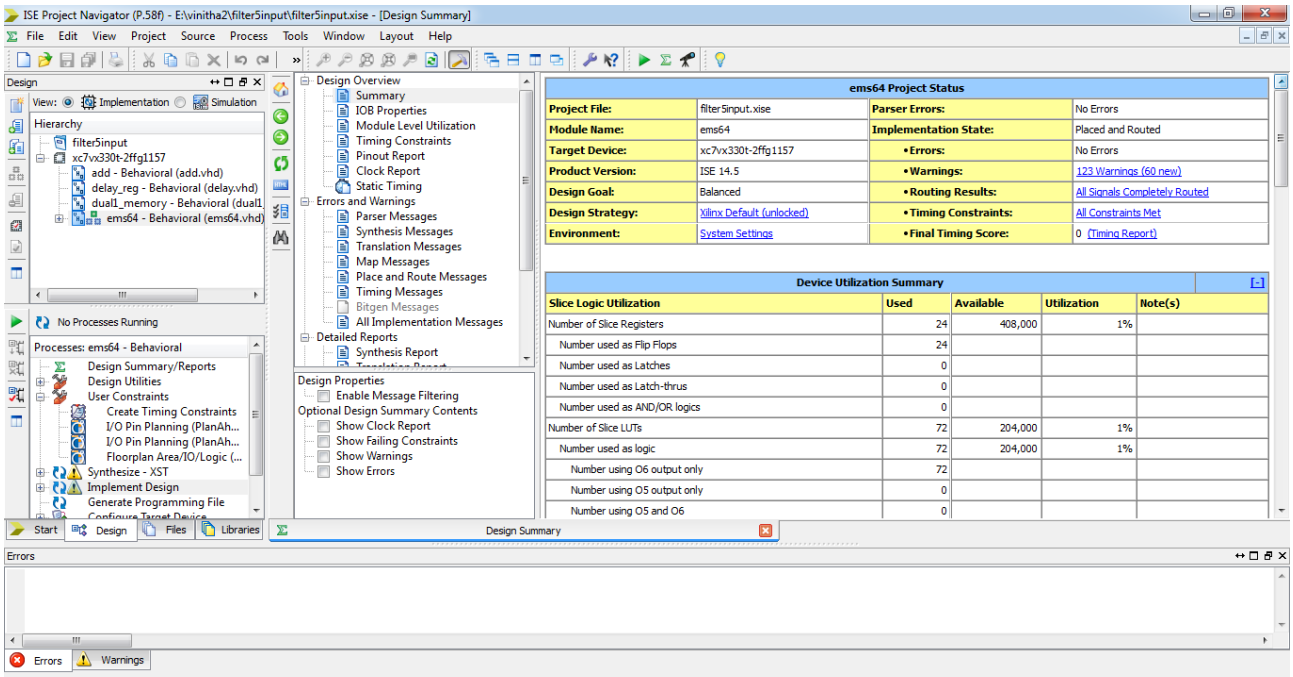

Fig. 8 Device utilization summary of EMS multiplier with 64-bit input.

product terms in memory. The variation comes in the external combinational circuit required in the different designs. Hardware complexity of the conventional LUT-based multiplier and the proposed designs and the OMS design of [28] are given in Table 8. For supporting our result, the device utilisation summary report of the tool is shown in Figs. 7 and 8 for OMS and EMS multiplier with 64-bit input.

Hence on comparison we can conclude that by storing either the even or odd product terms we are reducing the size of the memory by half which we achieved as that of the design proposed in [28]. But the over-head circuits used by our design to derive the odd or even multiplies is simpler which is obvious in the hardware utilisation result given by the tool. Also the maximum path delay of our design is very less as compared to the design proposed in [28]. As the length of the input increases, our design gives a better area-efficient memory-based

multiplier.

Thus the newness in the proposed LUT Design on FPGA for Memory-Based multiplication is as follows:

1. The memory space required to store the product terms in the case of memory-based multiplier is reduced to half by storing only the even product terms.
2. Proposed a simpler external circuit to derive the odd product terms.
3. The proposed multiplier when implemented in FPGA gives an area and speed efficient design.
4. The hardware complexity when compared with the already proposed memory-based multiplier is very less, which is well summarised in Table 6. Thus the memory-based multiplier design proposed by us proves an area-efficient design.
5. The maximum combinational path delay between the input and the output is very less because of the simpler external combinational circuit. As compared with the already proposed memory-based multiplier, the maximum path delay is decreased by 77%.

In the next section we demonstrate that by employment of the proposed LUT-based multiplier of this communication in the place of conventional memory-based multiplier of the FIR filter, the area-complexity of the filter can be improved without affecting the performance of the filter.

4 Memory-based FIR Filter Structure Using the Proposed EMS-based LUT Multiplier

Transposed FIR filter structure is considered because of the self-pipelined structure, that is, it has an optimum critical path which is equal to one multiplier and one adder time without adding any extra delay elements [3]. The modified transposed structure of FIR filter replacing each multiplier by a conventional LUT- based multiplier is given in Fig. 9. Conventional LUT-based multiplier consists of a dual core memory to store the product terms followed by shift adder to add the MSB and LSB part of the product term. Also the proposed designs of LUT-based multiplier best suits to this structure. Each adder operation is separated by delay elements in this structure and this type of structure is suitable to FPGA or ASIC implementation.

For an N-tap filter there will be N multipliers and (N-1) adders. In N multipliers constant coefficients are multiplied with the input data. In transposed form, the input data is common to all the N multipliers. Hence switching power is decreased. In Memory-based filter Structure, the N multipliers are replaced by the proposed EMS-based-LUT multiplier. Since the input data is common to all the multipliers, a common decoder circuit is used to address the entire N- Memory array, which is the part of the LUT-multiplier. The proposed EMS-based filter structure is shown in Fig. 10. For example, we have considered the input data length equal to 8-bit and the fixed coefficient word equal to W-bit. Two decoders D_1 and D_2 are used to decode the MSB [X_7 X_6 X_5 X_4] and LSB [X_3 X_2 X_1 X_0] part of the input data and address the dual product terms stored in the N dual core memories. The even product terms M_1 and M_2 fetched from the memory is fed to the adder-1 and adder-2 to derive the odd product terms. Both even and odd product terms are fed as inputs to mux-1 and mux-2. Mux M_1 and M_2 with the help of the select line which is connected to X_4 and X_0 bit of the input data respectively will select the corresponding product term and pass it on to the shift adder circuit. Shift adder shifts the MSB (M_1) product term left by 4-bits and adds the result with the LSB (M_2) product term. The output of the LUT multiplier is further delayed in delay block (D) and added in the adder (A) cell of the filter. The output of the filter is available after a latency of three clock pulses which are equal to one memory fetch cycle, one shift-add operation and the last equal to the adder operation of the filter. The actual output is available after (N+2) cycles because the output from first (N-1) cycles does not have the contributions from all the filter coefficients.

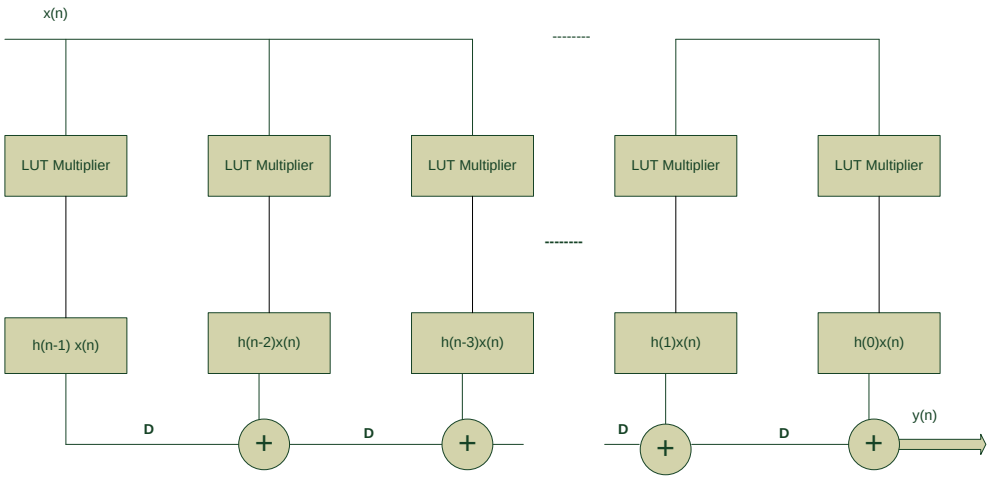

Fig. 9 Modified transposed FIR filter structure.

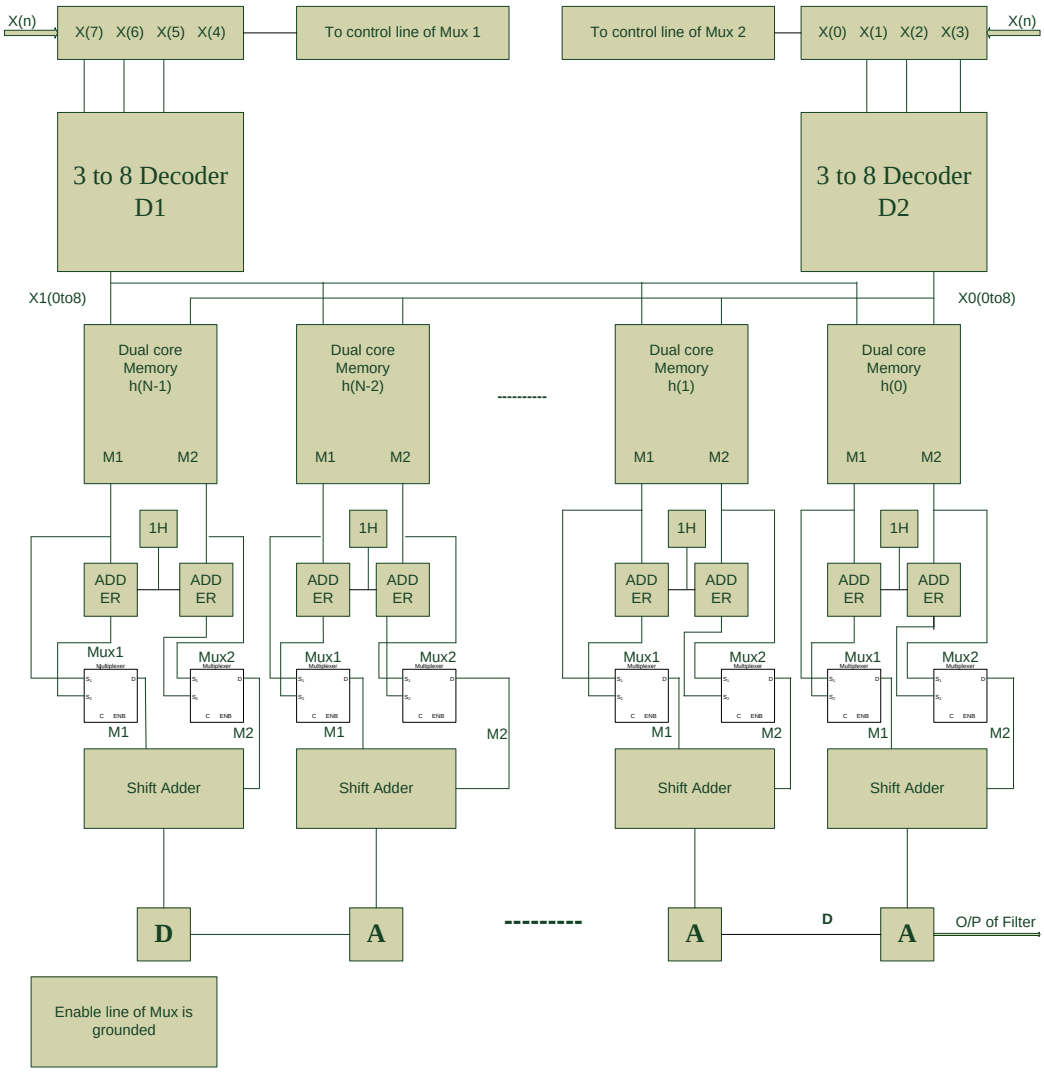

Fig. 10 Transposed FIR filter structure using the proposed EMS-LUT multiplier.

Similarly we can design filter using MOMS-LUT multiplier. The variation will be there in the product term (odd product terms) stored in the memory and in the place of adder, subtractor is used to derive the even product terms and the order of inputs applied to the mux will vary. But the latency and throughput of both the filter remains same. Both the filter structure and the OMS-LUT filter proposed in [28] are simulated and implement in FPGA and their hardware complexity and the maximum frequency of operation of the filter are compared.

The comparison of the hardware utilisation parameters for input word length 8 for different order of the filter is given in Table 9. Similarly, the comparison of Maximum frequency of operation for different word length and different order of filter is given in detail in Table 10. From Table 9, we can see that the number of slices and the number of slice LUT's require to implement both EMS and MOMS based filter is very less compared to the OMS based filter. From Table 10, we can infer that the maximum frequency of operation is higher for all filter orders and for various input data length for the proposed design. Since the FIR filter considered for implementation is a pipelined structure, apart from the initial latency of the filter which will be same for our design and the OMS design, the proposed filter works at a higher throughput rate compared to the OMS design.

5 Conclusion

Two new techniques are proposed namely, EMS-LUT multiplier and MOMS-LUT multiplier in contrast to an already existing technique OMS-LUT multiplier to reduce the area complexity and the path delay of a Memory-based multiplier. In both the techniques the size of the LUT is reduced by half. Also the complexity of the multiplier is reduced because of simpler logical circuits utilized to derive the odd product term in case of EMS and even product term in case of MOMS design. The proposed design is then tested with Xilinx tool and implemented in FPGA of type Virtex 7 XC7vx330tffg1157. For a 64-bit input we are

Table 9 Comparison of hardware utilisation parameters for the input word.

Order of the filter	Proposed design Area in terms of		OMS-LUT design [28] Area in terms of	
	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's
16	100	267	122	203
32	228	695	307	892
64	973	2915	986	3015

Table 10 Comparison of F_{max} for different input word length and different filter order.

Order of the filter	Proposed design Freq. [MHz] Input word size		OMS-LUT design [28] Freq. [MHz] Input word size	
	No. of slices	No. of slice LUT's	No. of slices	No. of slice LUT's
	[4]	[8]	[4]	[8]
16	433.46	357.52	327.22	200.60
32	304.87	303.85	191.31	254.38
64	228.93	198.49	206.44	173.82

able to achieve an area efficiency of 62% and maximum path delay efficiency of 77% in comparison to [28]. This proposed multiplier is then used in FIR filters and the complexity reduction in filter is verified. The maximum frequency of operation of the filter achieved is also higher in proposed multiplier design-based filter. The hardware utilised, as external control circuit for an *N*-th order filter by our proposed design are, *N*(2([*W*+8]/2))-bit adders, *N*(2(2 to 1)) mux. The hardware utilised by the OMS design [28] are *N*(2(4 to 3)) encoder, *N*(2(*W*+4))-bit NOR gates, *N*(2(*W*+4)-bit AOI gates, *N*(2(2-bit)) NOR gates, *N*(2(2-bit)) OR gates apart from dual core memory, decoder and shift adder which is used by both the designs. For a 32-order filter, filtering an input with a word length of 8, the area efficiency achieved using our multiplier is 26%. For any DSP application we require higher order filter and for better performance the word length of the input should be high. For higher order filtering with larger input length, our proposed filter is a better alternative compared to other optimised design proposed till now under memory-based filters.

References

[1] S. M. Kuo, *Real-time digital signal processing: implementations and applications*. Hoboken, NJ: John Wiley, 2006.

[2] S. Sheng, A. Chandrakasan, and R. W. Brodersen, "A portable multimediaterminal," *IEEE Communications Magazine*, Vol. 30, No. 12, pp. 64–75, 1992.

[3] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. New York: John Wiley & Sons Inc., 1999.

[4] S. K. Mitra, *Digital signal processing: a computer based approach. boston*: McGraw-Hill, 2006.

[5] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms and applications*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[6] M. D. Macleod and A. G. Dempster, "Multiplierless FIR filter design algorithms," *IEEE Signal Processing Letters*, Vol. 12, No. 3, pp. 186–189, 2005.

[7] D. L. Maskell, J. Leiwo, and J. C. Patra, "The design of multiplierless FIR filters with a minimum adder step and reduced hardware complexity," in *IEEE International Symposium on Circuits and Systems*, 2006.

[8] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Magazine*, Vol.9, No.3, pp.16–35, 1992.

[9] M. Kuhlmann and K. K. Parhi, "A high-speed CORDIC algorithm and architecture for DSP applications," in *IEEE Workshop on Signal Processing Systems*, pp. 732–741, 1999.

[10] S. A. White, "Applications of the distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, Vol. 6, No. 3, pp. 5–19, 1989.

[11] H. R. Lee, C. W. Jen, and C. M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," I*EEE Transactions on Consumer Electronics*, Vol. 39, No. 3, pp. 619–629, 1993.

[12] K. Itoh, S. I. Kimura, and T.Sakata, "VLSI memory technology: Current status and future trends," in *IEEE European Conference on Solid-State Circuits (ESSCIRC'99)*, pp. 3–10,1999.

[13] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. Mckenzie, "Computational RAM: Implementing processors in memory," *IEEE Transactions on Design & Test of Computers*, Vol. 16, No. 1, pp. 32–41, 1999.

[14] C. S. Vinitha and R. K. Sharma, "Memory-based VLSI architectures for digital filters: A survey," in *IEEE International Conference (UPCON-2016)*, pp. 98–101, 2016.

[15] M. Z. Zhang and V. K. Asari, "A fully pipelined multiplier-less architecture for 2-D convolution with quadrant symmetric kernels," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 1559–1562, 2006.

[16] Y. H. Chan and W. C. Siu, "On the realization of discrete cosine transform using the distributed arithmetic," *IEEE Transactions on Circuits and Systems I, Fundamental theory and applications*, Vol. 39, No. 9, pp.705–712, 1992.

[17] 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 Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 3, pp. 445–453, 2005.

[18] P. K. Meher, "Unified systolic-like architecture for DCT and DST using distributed arithmetic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 53, No. 5, pp. 2656–2663, 2006.

[19] J. I. Guo, C. M. Liu, and C. W. Jen, "The efficient memory-based VLSI array design for DFT and DCT," *IEEE Transactions on Circuits and Systems II-Analog and Digital Signal Processing*, Vol. 39, No. 10, pp. 723–733, 1992.

[20] D. F. Chiper, "A systolic array algorithm for an efficient unified memory-based implementation of the inverse discrete cosine transform," in *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, pp. 764–768, 1999.

[21] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad, and T. Stouraitis, "Systolic algorithms and a memory-based design approach for a unified architecture for the computation of DCT/DST/IDCT/IDST," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 52, No. 6, pp. 1125–1137, 2005.

[22] P. K. Meher and M. N. S. Swamy, "New systolic algorithm and array architecture for prime-length discrete sine transform," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 54, No. 3, pp. 262–266, 2007.

[23] P. K. Meher, J. C. Patra, and M. N. S. Swamy, "High-throughput memory-based architecture for DHT using a new convolutional formulation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 54, No. 7, pp.606–610, 2007.

[24] P. K. Meher, "Memory-based hardware for resource-constraint digital signal processing systems," in *Proceedings of IEEE 6th International Conference on Information, Communications & Signal*, pp. 1–4, 2007.

[25] P. K. Meher, "Low-latency hardware-efficient memory-based design for large-order FIR digital filters," in *Proceedings of IEEE International Conference on Information, Communications & Signal*, 2007.

[26] P. K. Meher, "New approach to LUT implementation and accumulation for memory-based multiplication," in *IEEE International Symposium on Circuits and Systems (ISCAS'09)*, pp. 453–456, 2009.

[27] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proceedings of IEEE International Symposium on Integrated Circuits (ISIC'09)*, pp. 663–666, 2009.

[28] P. K. Meher, "New approach to look-up-table design and memory-based realization of FIR digital filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 57, No. 3, pp. 592-603, 2010.

[29] P. K Meher, "LUT optimization for memory-based computation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 57, No. 4, pp. 285–289, 2010.

[30] P.K. Meher, "Memory-based computation of inner-product for digital signal processing applications," in *IEEE International Symposium on. Electronic System Design (ISED)*, pp. 95–100, 2010.

[31] P. K. Meher, "Novel input coding technique for high-precision LUT-based multiplication for DSP applications," in *IEEE. Conference on VLSI System on Chip (VLSI-SoC)*, pp. 201–206, 2010.

[32] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Area-delay trade-off in distributed arithmetic based implementation of FIR filters," in *Proceedings Tenth International Conference on VLSI Design*, pp. 124–129, 1997.

[33] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, No. 5, pp. 125–128, 2005.

[34] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 52, No. 7, pp. 1327–1337, 2005.

[35] S. S. Jeng, H. C. Lin, and S. M. Chang, "FPGA implementation of FIR filter using M-bit parallel distributed arithmetic," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2006.

[36] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," *IEEE Transactions on Signal Processing*, Vol. 56, No. 7, pp. 3009–3017, 2008.

[37] J. P. Choi, S. C. Shin, and J. G. Chung, "Efficient ROM size reduction for distributed arithmetic," in *IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No. 00CH36353)*, No. 2, pp. 61–64, 2000.

[38] P. K. Meher and S.Y. Park, "A novel DA-based architecture for efficient computation of inner-product of variable vectors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 369–372, 2014.

[39] Y. Pan and P. K. Meher, "Efficient coefficient partitioning for decomposed DA-based inner-product computation," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 406–409, 2011.

[40] C. S. Vinitha and R. K. Sharma, "A Novel Technique to optimize the LUT used in Memory based filter," in *IEEE International Conference in Electrical, Electronics, Computers, Communication, Mechanical and Computing*, 2018.

[41] C. S. Vinitha, R. K. Sharma, "New approach to low-area, low-latency memory-based systolic architecture for FIR filter," *Journal of Information and Optimisation Sciences*, Vol. 40, No. 2, pp. 247–262, 2019.

[42] M. T. Khan and R. A. Shaik, "Optimal complexity architecture for pipelined distributed arithmetic based LMS adaptive filter," *IEEE Transactions on Circuits and Systems I: Regular papers*, Vol. 66, No. 2, pp. 630–642, 2019.

[43] M.S. Prakash and R. A. Shaik, "Low area and high throughput architecture for an adaptive filter using distributed arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 60, No. 11, pp. 781–785, 2013.

[44] R. Guo and L. S. Debrunner, "Two high performance adaptive filter implementation scheme using distributed arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 58, No. 9, pp. 600–604, 2011.

[45] B. K. Mohanty and P. K. Mehar, "A high performance enery-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm," *IEEE Transactions on Signal Processing*, Vol. 61, No.4, pp. 921–932, 2013.

[46] Sang Yoon Park and Pramod K. Mehar, "Low power, high throughput and low area adaptive FIR filter based on distributed arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 60, No. 6, pp. 346–350, 2013.

[47] R. Kamal, P. Chandravanshi, N. Jain and R. Kumar, "Efficient VLSI architecture for FIR filter using DA-RNS," in *IEEE International conference on Electronics, Communication and Computational Engineering,(ICECCE)*, 2014.

[48] B. K. Mohanty, P. K. Mehar and S. K. Patel, "LUT optimisation for distributed arithmetic based block LMS adaptive filter," *IEEE Transactions on VLSI Systems*, Vol. 24, No. 5, pp. 1926–1935, 2016.

C. S. Vinitha did her B.Tech degree from Bharathiar University, Coimbatore, India and M.Tech from AIT, Indraprastha University, Delhi, India. Presently she is pursuing Ph.D. from IP University, Delhi. She is working as Assistant Professor in AIACT&R, Delhi. Her research interest includes signal processing, digital System design and VLSI design.

R. K. Sharma was born in Allahabad in 1964. He received his Diploma (Electronics Engineering) in 1984, AMIE (India) in 1989, M.E. (Control and Instrumentation) in 1994 and Ph.D. from University of Delhi in 2007. He has served as Lecturer in Ambedkar Polytechnic during 1997 to 2001, in Netaji Subhas Institute of Technology during 2001-2004, as Assistant Professor/Associate Professor in Ambedkar Institute of Advanced Communication Technologies and Research, Delhi during 2004-2013. Currently he is serving in the same institute as Professor since 2013. Currently he is holding the chair of Principal, Ambedkar Institute of Advanced Communication Technologies and Research since December, 2018. His areas of research interests are analog microelectronics and analog signal processing, mixed signal circuit design, circuit theory, VLSI design.