Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Research Paper

FPGA's Dual-Port ROM-Based 8x8 Multiplier for Area Optimized Implementation of DSP Systems

A. Pathan*,**(C.A.) and T. Memon***

Abstract: FPGA's block memory may be programmed as a single or dual-port RAM/ROM module that leads to an area-efficient implementation of memory-based systems. In this contest, various works of carrying out an optimized implementation of simple to complex DSP systems on embedded building blocks may be seen. The multiplier is a core element of the DSP systems, and in implementing a memory-based multiplier, it is observed that one of the operands is kept constant, hence leading the design to a constant-coefficient multiplication. This paper shows Virtex-7 FPGA's dual-port ROM-based implementation of an 8x8 variable-coefficient multiplier that may be used in several simple to complex DSP applications. The novelty of the proposed design is to configure the block ROM in dual-port mode and, hence, get four partial products in two clock cycles and introduce two unconventional adder approaches for partial product addition. This approach leads to fully resource utilization and the provision of a variable-coefficient multiplier. The work also shows the comparison of proposed architecture with already existing memory-based implementations and concludes the work as a novel step towards the efficient memory-based implementation of multiplier core.

Keywords: Block Memory, Digital Signal Processing, FPGA, Multiplier.

1 Introduction

THE FPGA may be used to implement computation-intensive algorithms more efficiently in comparison to DSP or microprocessors[1]. The three main factors that play an essential role in FPGA-based design are the targeted FPGA architecture, electronic design automation (EDA) tools, and design techniques employed at the algorithmic level using hardware description languages [2].

Iranian Journal of Electrical and Electronic Engineering, 2021.
Paper first received 12 October 2020, revised 08 April 2021, and accepted 25 April 2021.
* The author is with the School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, Australia.
** The author is with the Electronic Engineering Department, Quaid-e-Awam University of Engineering, Science & Technology, Larkana Campus, Pakistan.
E-mail: pathan_aneela@quest.edu.pk.
*** The author is with the School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, Australia and Department of Electronic Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan.
E-mail: tayabuddin.memon@faculty.muet.edu.pk.
Corresponding Author: A. Pathan.
https://doi.org/10.22068/IJEEE.17.4.2011

For example, Xilinx Virtex-7 FPGA, a selected device, contains several features and embedded DSP cores to strengthen its arithmetic capabilities along with dual-or, single-port RAM modules, ROM modules, synchronous FIFOs, and registers that may be easily implemented using the Xilinx CORE Generator [3].

The block RAM stores up to 36 kb of data and can be configured as either two independent 18 kb RAMs or one 36 Kb RAM. Each memory can be addressed through single ports and configured as a dual-port RAM/ROM.

Besides, the n-bit input Look-up table of an FPGA can be used for storing the truth table of an n-input function, or 2^n–bit data consequently. For example, a LUT with three inputs can store the truth table of any 3-input function or an eight-bit word.

Taking the benefit of this architectural flexibility, much work may be seen in the domain of memory-based (Look-up tables (LUT), BRAM) DSP systems design, especially the multiplier.

The multiplier is a core element of the DSP system, and in implementing a memory-based multiplier, it is observed that one of the operands is kept constant, hence leading the design to constant-coefficient multiplication.

This paper shows Virtex-7 FPGA's dual-port ROM-based implementation of an 8x8 variable-coefficient multiplier that may be used in several simple to complex DSP applications.

The proposed design's novelty is to configure the block ROM in dual-port mode and hence get a total of four partial products in two clock cycles and introduce two unconventional adder approaches for partial product addition.

This approach leads to fully resource utilization and the provision of a variable-coefficient multiplier.

Besides, the comparison of proposed architecture with already existing memory-based implementations is also given, and in conclusion, it is shown that the proposed work is a novel step towards the efficient memory-based implementation of multiplier core.

The paper further proceeds as follows. In Section 2, previous work done in the domain of memory-based multiplier is given, followed by the proposed design in Section 3. The FPGA-based implementation and results are given in Section 4, and the conclusion and future work are reported in Section 5.

2 Previous Works

The FPGA-based implementation of a memory-based multiplier is possible in two ways 1) using the RAM/ROM module, 2) The look-up table-based design.

The two common approaches in LUT-based memory design are: using Direct-LUT to compute the multiplication [4-10] and to compute the inner-product using Distributed Arithmetic (DA) [11-15].

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, the multiplication is done [16].

In DA-based computation, the N-point vector's inner-product with the N-bit vector is pre-computed and stored in LUT [11].

LUT's size 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.

As the memory-based multiplier requires an adequate amount of memory of size $2^{2L,}$ where L is the word length of the operands, most of the work in the research leads to the design of a fixed-coefficient multiplier (leading to memory reduction from 2^{2L} to 2^{L}).

One of the earlier techniques to implement fixed-coefficient multipliers using Look-up table-based memory of FPGA was developed by Xilinx [17]. This relies on look-up tables rather than a network of adders to perform most of the multiplication.

As it is evident that there are sixteen possible results when a four-bit number is multiplied by an eight-bit fixed number (because there are sixteen different four-bit numbers). Thus a four-bit variable time eight-bit constant multiplier can be implemented by a sixteen-entry look-up table. Each entry must be twelve-bits (the width of the largest possible output). This idea may easily be understood as follows: Let the eight-bit constant multiplier is 250_{10} (11111010_2), and four-bit variable multiplicands are 0-15(0000-1111). So the LUT entries are shown in Table 1.

With the same approach, an eight-bit by eight-bit constant multiplier may be built using two of this four-bit by eight-bit constant multipliers in the configuration shown in Fig 1.

Besides LUT-based implementation, another way to get the same results is to use FPGA's built-in memory (RAM/ROM) modules.

In [17] and [19], an eight-bit constant multiplier is implemented using the Xilinx 4010 FPGA's Block-ROM. The designed architecture is shown in Fig. 2.

As in a 4X8 multiplier, we need a total of 16-LUTS, each twelve-bit wide; the same is required for RAM/ROM-based multiplier. The output of each memory is twelve-bit wide, but to save some of the resources, an eight-bit-by twelve-bit adder is used to add two outputs and then concatenates the results with the remaining for-bits to get the total product sixteen-bit wide.

Table 1 LUT values for a constant multiplier (250_{10}).

MP*	MC*	PR*	12-bit-value stored in LUT
	0 (0000)	0	LUT0=000000000000
	1 (0001)	250	LUT1=000011111010
	2 (0010)	500	LUT2=000111110100
	3 (0011)	750	LUT3=001011101110
	4 (0100)	1000	LUT4=001111101000
	5 (0101)	1250	LUT5=010011100010
	6 (0110)	1500	LUT6=010111011100
250	7 (0111)	1750	LUT7=011011010110
	8 (1000)	2000	LUT8=011111010000
	9 (1001)	2250	LUT9=100011001010
	10(1010)	2500	LUT10=100111000100
	11(1011)	2750	LUT11=101010111110
	12(1100)	3000	LUT12= 101110111000
	13(1101)	3250	LUT13=110010110010
	14(1110)	3500	LUT14=1101101011100
	15(1111)	3750	LUT15= 111010100110

*MP = Multiplier, *MC = Multiplicand, *PR = Product value.

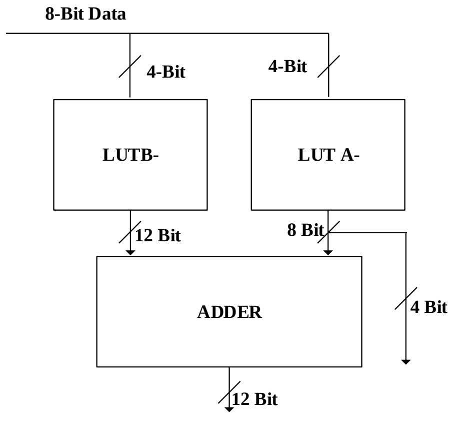

Fig. 1 Basic LUT-based constant multiplier [18].

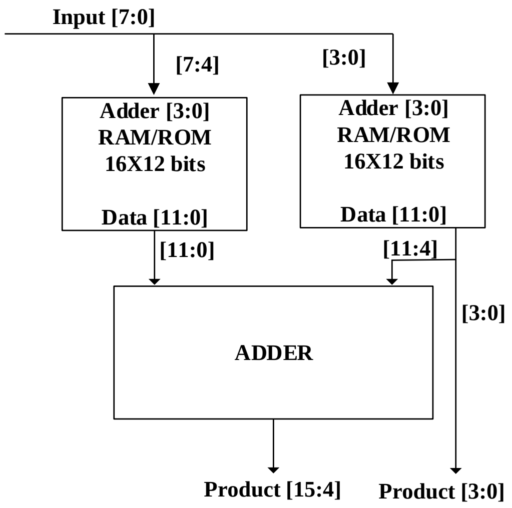

Fig. 2 Basic architecture of an eight-bit constant multiplier using memory [19].

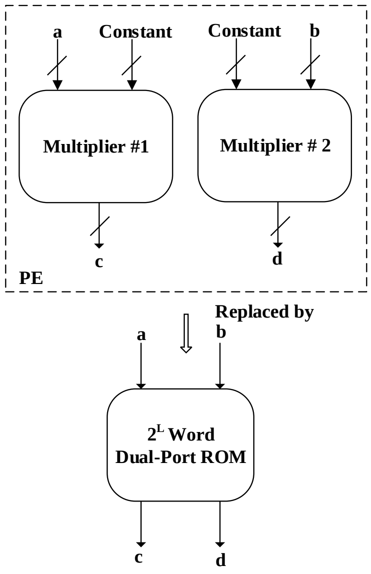

Fig. 3 Dual-port ROM based implementation of N-bit-by-N-bit multiplier [4].

If one of the two operands is fixed, a significant reduction in the memory size from 2^{2L} to 2^L is obtained. This idea is reflected in [4], where the dual-port ROM is used to replace two constant multipliers (any N-bit multiplier may be designed using two N/2-bit-by-N/2 multipliers and then adding the partial products to get the N-bit multiplier).

Fig. 3 shows an N-bit-by-N-bit memory-based implementation [4].

The work given in [6] shows more efficient memory-based systolic array implementation of the unified architecture of DCT/DST/IDCT/IDST, using dual-port ROMs and appropriate hardware sharing methods.

In design (Fig. 4), the conventional constant multiplier is replaced with two dual-port ROMs, each of size of $2^{2L/2}$.

In contrast to the efficient utilization of selected FPGA's resources, a plausible work also reports an optimized design with algorithm optimization.

For example, in DA-based computation, offset binary coding [11, 15], and group distributed technique [13] are proposed to decrease the size of the memory. Whereas under Direct-LUT based computation, many techniques that are proposed are given in [10, 20, 21].

Furthermore, in [10, 22], authors have proposed the OMS approach, where only the odd multiple product terms are stored in memory. Thus the size of the memory is reduced by half. Similarly, in [23, 24], another technique, namely, Anti-symmetric Product Coding (APC), has been detailed, where the size of the LUT is reduced again. Also, it requires fewer overhead circuits as compared to the work of [10].

In [25], authors have combined both OMS and APC techniques and developed an efficient architecture that contains the advantages of both of the above techniques.

In the most recent work reported in [26], two

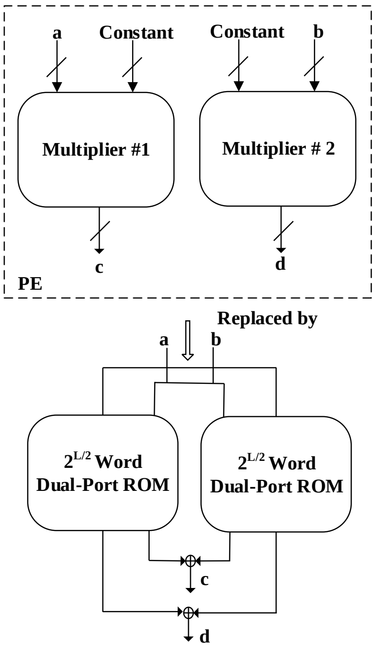

Fig. 4 A reduced word-length multiplier replaced by a dual-port ROM [6].

memory-based multipliers are proposed and implemented on Vertex 7 XC7vx330tffg1157 FPGA. The first technique is EMS-LUT based multiplier, where even multiplies of the product terms are stored in memory (a single-port ROM to store the pre-calculated even terms of the product value of four-bit input), where an external combinational logic circuit is used to derive the odd multiplies of the product term. The second approach, Modified OMS-based multiplier, is the variant of already existing OMS multiplier design [21] in which some changes are brought in the external control circuit used to derive the even product terms.

All the research in one way or another covers a

$$
\begin{array}{ccccc}
 & M_1 & M_0 & & \\
\times & N_1 & N_0 & & \\
\hline
 & M_1N_0 & M_0N_0 & PP_1 & \\
M_1N_1 & M_0N_1 & \times & PP_2 &
\end{array}
$$

Fig. 5 The conventional multiplication method.

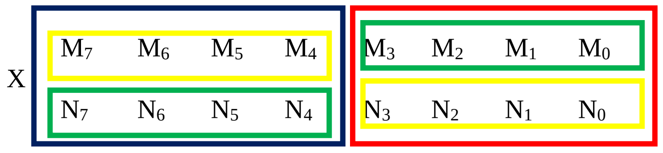

Fig. 6 Division of product into various parts.

			M_3	M_2	M_1	M_0	
	X		N_3	N_2	N_1	N_0	
A			M_3N_0	M_2N_0	M_1N_0	M_0N_0	pP1
		M_3N_1	M_2N_1	M_1N_1	M_0N_1	X	pP2
	M_3N_2	M_2N_2	M_1N_2	M_0N_2	X	X	pP3
M3N3	M_2N_3	M_1N_3	M_0N_3	X	X	X	pP4

Fig. 7 Product of [3:0] bits of multiplier with [3:0] bits of the multiplicand.

			M_7	M_6	M_5	M_4	
	X		N_3	N_2	N_1	N_0	
B			M_7N_0	M_6N_0	M_5N_0	M_4N_0	pP1
		M_7N_1	M_6N_1	M_5N_1	M_4N_1	X	pP2
	M_7N_2	M_6N_2	M_5N_2	M_4N_2	X	X	pP3
M_7N_3	M_6N_3	M_5N_3	M_4N_3	X	X	X	pP4

Fig. 8 Product of [3:0] bits of multiplier with [7:4] bits of the multiplicand.

			M_3	M_2	M_1	M_0	
	X		N_7	N_6	N_5	N_4	
C			M_3N_4	M_2N_4	M_1N_4	M_0N_4	pP1
		M_3N_5	M_2N_5	M_1N_5	M_0N_5	X	pP2
	M_3N_6	M_2N_6	M_1N_6	M_0N_6	X	X	pP3
M_3N_7	M_2N_7	M_1N_7	M_0N_7	X	X	X	pP4

Fig. 9 Product of [7:4] bits of multiplier with [3:0] bits of the multiplicand.

			M_7	M_6	M_5	M_4	
	X		N_7	N_6	N_5	N_4	
D			M_7N_4	M_6N_4	M_5N_4	M_4N_4	pP1
		M_7N_5	M_6N_5	M_5N_5	M_4N_5	X	pP2
	M_7N_6	M_6N_6	M_5N_6	M_4N_6	X	X	pP3
M_7N_7	M_6N_7	M_5N_7	M_4N_7	X	X	X	pP4

Fig. 10 Product of [7:4] bits of multiplier with [7:4] bits of the multiplicand.

constant-coefficient multiplier and indicate that the design complexity and hardware requirement increases with an increase in the word length of a multiplier (or keeping it variable as original).

To get out of this limitation and effectively utilizing the available resources, this paper shows Virtex-7 XC7vx330tffg1157 FPGA's dual-port ROM-based implementation of an 8x8 unsigned integral multiplier that may be used in several simple to complex DSP applications.

3 Proposed Design

In a memory-based multiplier, pre-calculated product values are easy to obtain within the minimal processing time, as they are stored at particular addresses masked with operands [27].

In general, an individual memory module is needed to store the data values obtained by multiplying a constant operand (multiplier) with several other operands (multiplicands) depending on the word length.

For understanding the concept, lets us take a 2x2 multiplier. Here the word length (in bits) of both operands is 2, and possible two-bit values in decimals are 0, 1, 2, and 3.

This detail shows that a total of 2^n operands are possible with n-bit wide data. Hence, for an operand of the same length, the total number of memory modules needed to store the product values are 2^{2n} with an individual size be as a 2^nx2^n array leading the product length to NxN bit wide [5].

As the word length increases, the required memory also increases in proportion. Therefore, for area-efficient implementation, it is needed to cut down memory need to some extent and get required functional verifications possible with distributed arithmetic and efficient utilization of available resources.

Taking the benefit of these two techniques, in our design, we have used only a single block ROM module in its dual-port configuration along with some intermediate addition. The multiplier and multiplicand size is set to 8x8 that in the conventional approach leads to 65025 memory modules each of size 8x8 array with sixteen-bit long product value or a total of 127 Kb memories as an individual.

The design strategies of this work are taken from the concept of a conventional array multiplier. For a simple 2x2 multiplier, the concept is as follows.

Let M, N be two-bit operand, leading M_0, N_0 as LSB and M_1, N_1 be MSB bits of multiplicand and multiplier consecutively. PP1 and PP2 are two partial products as an intermediate stage, and these partial products PP1 and PP2 are then added to get the final result.

It may be observed that in the case of two-bit operands, we need one full-adder cascaded to one-half adders for summing PP1 and PP2.

If this concept is extended to an 8x8 multiplier, we need eight partial products PP1, PP2… PP8, and one 9-input 15-bit adder to add PP1-PP8; hence the circuit becomes a bit complex.

The same result may be obtained if we reduce the operand length from 8x8 to 4x4 and then perform some intermediate arithmetic [20]. Hence, the ROM size necessary to replace a multiplier can be further reduced at the cost of an extra adder [4]. Let us divide the multiplier and multiplicand, as shown in Fig.6.

A total of 4–4x4 multiplications would be performed in between the operand, as codded above. If A, B, C, and D show those multiplications, then individual may

Table 2 Memory configuration of BROM.

	Op1	1	2	3	…..	13	14	15
Op2	0	1	2	3	…..	13	14	15
0	0	0	0	0	…..	0	0	0
1	0	1	2	3	…..	13	14	15
2	0	2	4	6	…..	26	28	30
3	0	3	6	9	…..	39	42	45
4	0	4	8	12	…..	52	56	60
5	0	5	10	15	…..	65	70	75
….	….	….	….	….	….	…..	…..	….
11	0	11	22	33	…..	143	154	165
12	0	12	24	36	…..	156	168	180
13	0	13	26	39	…..	169	182	195
14	0	14	28	42	…..	182	196	210
15	0	15	30	45	…..	195	210	225

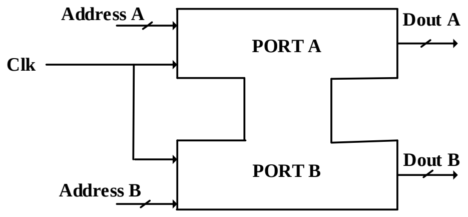

Fig. 11 General dual-port ROM configuration.

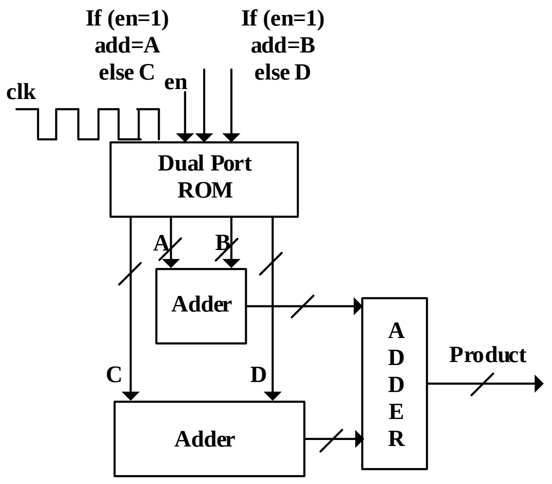

Fig. 12 Proposed architecture of memory-based multiplier.

be written as given in Figs. 7-10.

In dual-port block ROM, product values of a 4x4 multiplier as 16x16 array of eight-bit word length are pre-stored; this leads to 0.25 Kb of total memory.

Table 2 show the memory configuration for dual-port block ROM, where Op1 and Op2 show two operands (multiplier, multiplicand).

In general (Fig. 11), two addresses, each N bit wide, may be given as an input to BROM with a standard clock, and two outputs may be achieved.

In our proposed design (Fig. 12), the address values are created as per partial product generation, as shown in Figs. 7-10.

Two outputs are possible to achieve in a single time, so enable pin is used to select the two addresses amongst the four. Each address is 8-bit wide (four bits from the multiplier and four from multiplicand) and is given at port: Address A and Address B consecutively, and we get two pre-calculated product values at ports Dout A and Dout B.

As we deal with the synchronized clock approach, two clock cycles are required to get the output port's data (Fig. 13).

At the first clock cycle, addresses are given at address ports, and on the second, the data is taken at the output and at the same time (at second clock cycle), the addresses of ports are replaced with the new values, and finally, at third cycle, two other outputs are taken.

The partial products generated are then added after shifting the values for a particular number of bits. We need three adders for summing product A with B, C with D, and a final summation of those intermediated generated values.

Three approaches for partial product addition are followed separately to find out efficient implementation.

In the first design, a 3-input look-up table-based adder (Fig. 14) is used that obeys the conventional serial adder approach.

Three input look-up tables are defined as truth table given in Table 3 for sum out and carry out accordingly.

In the second implementation, the carry-look ahead adder is selected to sum the partial products, and in the third and final approach, FPGA's built-in arithmetic core is instantiated to perform the required arithmetic.

4 FPGA-Based Implementation and Results

The dual-port ROM-based proposed multiplier's architecture is implemented using Xilinx Virtex 7 XC7vx330tffg1157 FPGA and ISE 14.2 tool.

Lacking in finding a variable coefficient memory-based multiplier, this design is compared with an eight-bit constant-coefficient multiplier reported in [26].

To make comparison easy, some prevalent factors are considered; those may give a good analysis of consumed resources and efficiency of the system performance.

For example, the consumed slice count that contains logic elements (the look-up tables) used for the control circuitry and distributed arithmetic in a memory-based system design tells about FPGA area consumption. The block ROM is for storing the pre-calculated product values, multiplexers, adder/subtractors, decoders, and the most important performance parameters, the observed delay, and the maximum achieved frequency.
Table 4 shows the proposed design's implementation results with three different adder approaches, and the two memory-based constant-coefficient designs reported in [26].

As the constant-coefficient multiplier obviously would result in less resource utilization than a variable-coefficient multiplier, a one-to-one comparison cannot

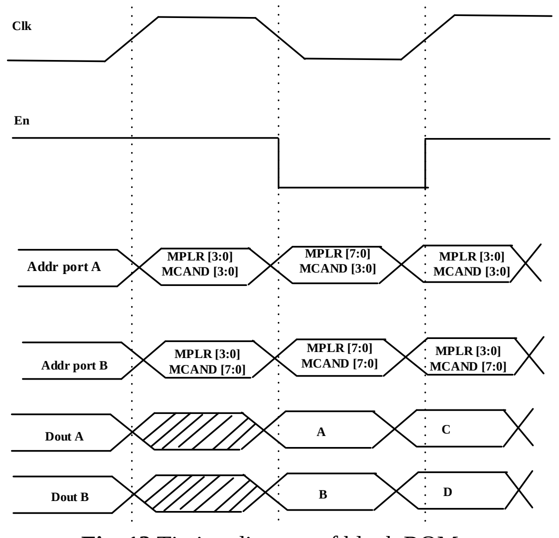

Fig. 13 Timing diagram of block ROM.

Table 3 Truth table for sum and carryout.

Cin	X	Y	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

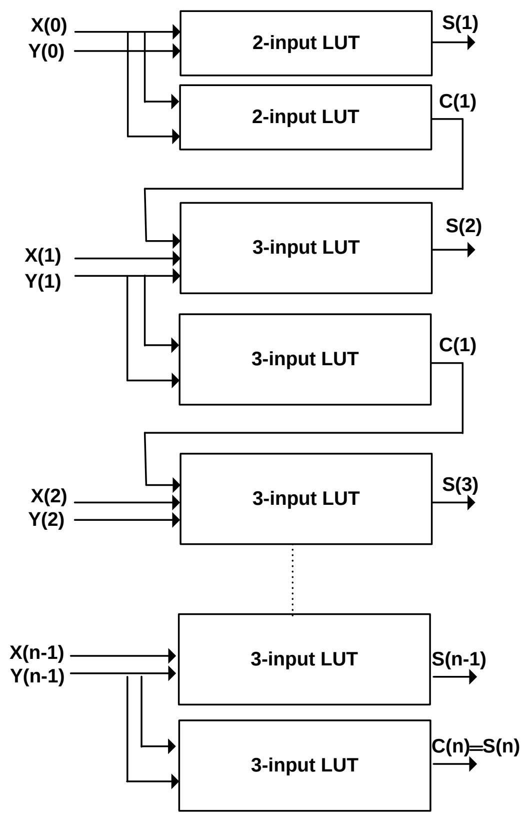

Fig. 14 Series configuration for n-bit LUT-based adder.

Table 4 FPGA-based results of proposed and conventional design.

Fact:	Prop: LUT-based design	Prop: CLA-based design	Prop: built-in adder-based design	EMS [26]	MOMS [26]
Slice	61	60	33	4	4
Add/Sub	0	0	1: 12 bit add: 2: 8 bit add:	1:[W+8] bit add: 2:[w+8/2] add:	1:[W+8] bit add: 2:[w+8/2] sub
RAM	1:8[256]	1:8[256]	1:8[256]	1:8[w+4]	1:8[w+4]
Deco	0	0	0	2:[3:8]	2[3:8]
Mult:	4:[2 to1]	4:[2 to 1]	4[2 to1]	2:[2 to 1]	2[2 to 1]
Delay	8.183	1.800	1.356	0.339	0.339
Freq.	122.206	555.41	737.456	2949	2949

be carried out. Alternatively, some analysis may be built based on some ratio between two implementation alternatives. Like, the number of slices in the proposed built-in adder-based design is 33 that is 8(4) times in EMS and MOMS-based design, hence approximately showing the same number if EMS and MOMS are designed for an 8x8 variable–coefficient multiplier.

Another factor to compare is the number or arithmetic primitives consumption. It is evident that the proposed LUT-based and CLA-based design even do not require any adder or subtractor hence needing no built-in IPCore for arithmetic.

The most crucial factor to consider is the amount of memory consumption. In our proposed design, only a single ROM of size 8x256 is required, whereas EMS and MOMS require 8xw+4 memory for a constant-coefficient multiplier that obviously will be 8(8xw+4) in case of a variable-coefficient multiplier. This analysis shows the proposed design to be very good in less consumed memory resources.

Similarly, no decoder is required in the proposed design, whereas the need is evident in EMS and MOMS-based design.

The number of multiplexers in the proposed designs is twice to that of their counterpart but still reflects a less number (that is probably four times greater in variable-coefficient multiplier).

As the speed is related to word length and overall circuit complexity, so the achieved frequency (reciprocal to delay observed) of our proposed design is far lesser than the work reported in [26], but once the conventional constant-coefficient based implementation is translated to a variable-co-efficient multiplier a sufficient reduction in achieved frequency would be observed.

5 Conclusion and Future Work

This work shows a dual-port Rom based area optimized implementation of an 8x8 multiplier using Virtex 7 XC7vx330tffg1157 FPGA and ISE 14.2 tool.

As it is already discussed, in all the implementations in memory-based multiplier design, one of the operand values is kept constant, resulting in some application limitations. Keeping this into view, our proposed design may be used in most of the DSP systems, needing variable coefficient multipliers, especially in image processing or in the domain of adaptive signal processing.

Besides, if one needs optimized area implementation and good achieved frequency, the carry-look ahead adder performs very well for intermediate addition.

Hence in the future, the feasibility of using dual-port ROM and carry-look ahead adder in more complex DSP system designs will be seen and implemented.

Acknowledgment

The Higher Education Commission (HEC), Pakistan, under the National Research Program for Universities (NRPU) grant Number 8521, National Center of Robotics, and Automation (NCRA) joint lab titled "Haptics, Human Robotics and Condition Monitoring Systems (HHCMS) Lab" established at Mehran University of Engineering and Technology, Jamshoro provided support to this research.

References

[1] T. J. Todman, G. A. Constantinides, S. J. Wilton, O. Mencer, W. Luk, and P. Y. Cheung, "Reconfigurable computing: architectures and design methods," *IEE Proceedings-Computers and Digital Techniques*, Vol. 152, pp. 193–207, 2005.

[2] M. H. Rais, "Efficient hardware realization of truncated multipliers using FPGA," *International Journal of Applied Science*, Vol. 5, pp. 124–128, 2009.

[3] U. Xilinx, "Series FPGAs memory resources: User guide, Report, Xilinx Inc," San Jose, 2015.

[4] J. I. Guo, C. M. Liu, and C. W. Jen, "The efficient memory-based VLSI array designs for DFT and DCT," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 39, pp. 723–733, 1992.

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

[6] D. F. Chiper, M. 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, pp. 1125–1137, 2005.

[7] P. K. Meher and M. Swamy, "New systolic algorithm and array architecture for prime-length discrete sine transform," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 54, pp. 262–266, 2007.

[8] P. K. Meher, J. C. Patra, and M. Swamy, "High-throughput memory-based architecture for DHT using a new convolutional formulation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 54, pp. 606–610, 2007.

[9] P. K. Meher, "Low-latency hardware-efficient memory-based design for large-order FIR digital filters," in *6th International Conference on Information, Communications & Signal Processing*, pp. 1–4, 2007.

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

[11] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, Vol. 6, pp. 4–19, 1989.

[12] 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, pp. 705–712, 1992.

[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 transactions on Circuits and Systems for Video Technology*, Vol. 15, pp. 445–453, 2005.

[14] 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, pp. 2656–2663, 2006.

[15] 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)*, pp. 61–64, 2000.

[16] H. R. Lee, C. W. Jen, and C. M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Transactions on Consumer Electronics*, Vol. 39, pp. 619–629, 1993.

[17] K. Chapman, "Fast integer multipliers fit in FPGAs (EDN 1993 design. idea winner)," *EDN Magazine*, May 1994.

[18] T. Kean, B. New, and B. Slous, "A fast constant coefficient multiplier for the XC6200," in *International Workshop on Field Programmable Logic and Applications*, pp. 230–236, 1996.

[19] R. J. Petersen and B. L. Hutchings, "An assessment of the suitability of FPGA-based systems for use in digital signal processing," in *International Workshop on Field Programmable Logic and Applications*, pp. 293–302, 1995.

[20] P. K. Meher, "Novel input coding technique for high-precision LUT-based multiplication for DSP applications," in *18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, pp. 201–206, 2010.

[21] C. Vinitha and R. 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.

[22] 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, pp. 592–603, 2009.

[23] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proceedings of the 12th International Symposium on Integrated Circuits*, pp. 663–666, 2009.

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

[25] P. K. Meher, "LUT optimization for memory-based computation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 57, pp. 285–289, 2010.

[26] C. Vinitha and R. Sharma, "An efficient LUT design on FPGA for memory-based multiplication," *Iranian Journal of Electrical and Electronic Engineering*, Vol. 15, No. 4, pp. 462–476.

[27] T. D. Memon and A. Pathan, "An approach to LUT based multiplier for short word length DSP systems," in *International Conference on Signals and Systems (ICSigSys)*, pp. 276–280, 2018.

A. Pathan received degrees of B.E. in Telecommunication from Mehran UET, Jamshoro, Sindh, Pakistan in 2008 and M.E. in Electronics Engineering from NED University Karachi in 2010. Currently, she is pursuing her Ph.D. degree (FPGA-based DSP system design) from Mehran UET. Mrs. Pathan worked as Assistant Manager (satellite receivers' design) in SUPARCO from March 2008 to February 2013. Since then she serves as Assistant Professor in Quaid-Awam University College of Engineering Science and Technology Larkana.

T. D Memon received a B.E. (Hons.) Electronics Engineering (First Class) and a P.G. Diploma Telecommunication and Control Engineering (First Class) from Mehran University of Engineering & Technology, Jamshoro, Pakistan, in 2003 and 2006, respectively. He received Ph.D. from Royal Melbourne Institute of Technology (RMIT), Melbourne, Australia in 2012. Currently, he is working as an Associate Professor in the Department of Electronic Engineering, MUET. His research interests include short word length DSP Systems, embedded systems, and their FPGA-based implementation.