Proceedings of the Fifth International Conference on Inventive Computation Technologies (ICICT 2022)
IEEE Xplore Part Number: CFP22F70-ART; ISBN:978-1-6654-0837-0

2022 International Conference on Inventive Computation Technologies (ICICT) | 978-1-6654-0837-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICICT54344.2022.9850686

FPGA Implementation of High Performance Precise Signed and Unsigned Multiplier using Ternary 6-LUT Architecture

Palle Mahendra
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India
pallemahendra1996@gmail.com

Ramesh S R
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India
sr_ramesh@cb.amrita.edu

***Abstract*—Recent applications using arithmetic computations such as multiplication and division are often used in video/image processing and ML (machine learning). DSP blocks that are available on FPGAs are high-performance multipliers that may be used to accelerate computation. These multipliers are less and have fixed placement on FPGAs. Also, they generate additional routing delays which are inefficient for lower bit width multiplications. This results in greater power consumption. Soft IP cores that are specifically intended for multiplication are supplied as an optional feature by FPGA vendors to their customers. It improves the performance while using less power, but these IP cores need further improvement. This has led to the development of general low-latency, reduced area, and accurate soft-core multiplier designs based on the architectural properties of FPGAs, such as lookup table (LUT) structures and rapid carry chains. This work aims to build accurate and approximate signed and unsigned multipliers for 8-bit configurations. It utilizes a single LUT5 with multiplexers instead of double LUT5 in LUT6 architecture. To sum it up, this architecture was designed in Verilog HDL and synthesized in Xilinx software. In this work, for a signed multiplier, a notable decrease in area and delay by 42.69% and 11.79% was observed and for an unsigned multiplier, 43.8% and 6.61% reduction were attained.**

***Index Terms*—Approximate Multiplier, Accurate Multiplier, LUT (Lookup table), DSP (Digital Signal Processing) Blocks, Ternary adder.**

I. Introduction

Digital signal and image processing rely heavily on the arithmetic operation of multiplications. For certain applications, the high-level performance of these Digital Signal Processing blocks could make their use inefficient in terms of both total attainment and area. Digital circuits must be more energy-efficient to meet the needs of today's microprocessors. These include machine learning, digital signal processing for multimedia and data mining, as well as automatic data recognition. If you're looking to save on power, speed, and area, then approximation computing is a great option. Adders and multipliers, which are among the most energy-intensive digital blocks, have been the primary focus of research on hardware-level approximation. Approximate multipliers have been suggested using a variety of techniques and methods. To produce n x n multipliers, approximate recursive multipliers rely on fundamental 2 x 2 approximate multiplier modules. Reed-Solomon encoder implementations based on DSP have a greater latency due to the delay routing induced by the placement of designated Digital Signal Processing blocks [1]. Manual Floor planning may be an option for small applications looking to improve their overall performance. To put it another way, a significant portion of the available DSP blocks (56 percent) are used in the JPEG encoder implementation. Many of these applications use up the DSP blocks on an FPGA, rendering them inaccessible to other applications that need them, and hence requiring LUT multipliers for those applications. Because of this, the orthogonal strategy of using DSP blocks and logic-based soft multipliers is critical for improving overall performance in various implementation circumstances. As a result, both Xilinx and Intel provide soft multipliers that are suitable for all logic-based signal processing and image processing applications [2].

The performance of the radix 4 multiplier's area-efficient booth encoding approach was improved by adding six input LUTs and linking it to a carry chain of Xilinx FPGAs. They don't employ partial product compressor trees, and their critical path lag time is substantial. Multiplying algorithms developed by Booth and Baugh Wooley are employed in Xilinx FPGAs in a space-efficient way. ALMs (adaptable logic modules) of just five lengths are used to create carry chains to decrease the active length of carry chains. As a result of the implementation, the FPGA resources are underutilized. As a result, Xilinx's current generation of FPGAs cannot be used to achieve this feature of confining the carry chain to ALMs of count five without compromising on requirements. Despite this, their proposed architecture of GPCs (Generalized Parallel Counters) does not employ the most efficient LUTs in two consecutive ALMs [3].

In distinction to conventional multipliers such as the serial, shift-and-add and, serial/parallel multipliers, which use just a little amount of space, they have significantly large critical path delays. The following are examples of possible multiplier design choices. For parallel multipliers based on the Wallace

and Dadda designs to achieve the lowest possible output latencies must have a wide surface area. Based on the features of the Booth and Wallace/Dadda multiplier and Vedic multipliers approaches, it has proposed for ASIC (Application Specific Integrated Circuit) based devices a very fast advanced multiplier design that employs Radix-4 recoding. The interpretation of these IP soft multipliers may be enhanced by using proper ways for encoding and reducing partial products, among other things. For example, as compared to the attainment of the multiplier described, our recommended right multiplier designmay slash the crucial route time by up to 52 percent, accordingto our calculations. Because of their built-in resilience to approximation-invoked errors, these applications are capable of delivering legitimate results even if a portion of the input data or modest calculations is erroneous or approximate [4]. Look for instances of this kind of application in image and signal processing, machine learning, and a broad variety of other probabilistic approaches, to name a few areas of study. Accurate and approximate multipliers may both be employed for hardware acceleration in such applications, allowing for area minimization and performance efficiency to be achieved. It is possible to create appropriate algorithms for approximation adders and multipliers that have varied performance gains by applying the notions of approximate computing tothe problem [5]. Section II presents the proposed method of LUT 6 Configuration using LUT5 with Multiplexer. Section IIIdescribes the Proposed LUT 6-based Approximate Unsigned multiplier using FPGA, and Section IV presents the Proposed LUT 6-based Approximate Signed multiplier design with FPGA. Section V discussed the Results and Implementation of the proposed signed and unsigned approximate multiplier on FPGA. Section VI brings the study to a conclusion.

II. PROPOSED LUT 6 SLICE STRUCTURE

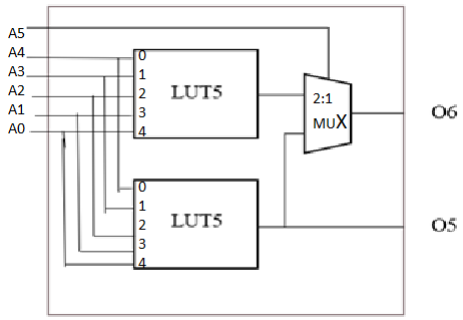


Fig. 1. The Architecture of LUT6 using DUAL LUT5s [1]

Xilinx and Intel's FPGAs, which are among the most advanced on the market, use LUT6 (six input LUT) to create combinational and sequential circuits. All of the designs presented in this book were implemented using Xilinx FPGAs, which we developed specifically for this purpose [6]. Although our approach is general, it may be executed on FPGAs from other suppliers, such as Intel, which employs LUT6 and carry chains in addition to ours. Four LUT6s are included in the customizable logic block (CLB) of Xilinx's 7-series FPGAs [7]. There are also 8 flip-flops (FFs) for recording LUT outputs and a single 4-bit length carry chain in each slice of the CLB.

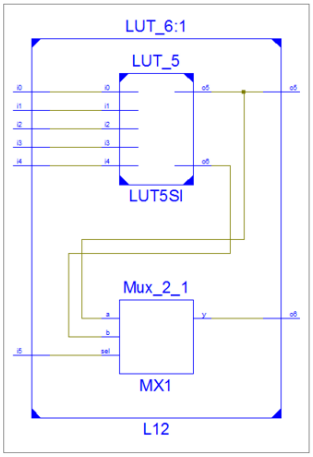


Fig. 2. The Architecture of the proposed LUT6 using only one LUT5 and Multiplexer

In Fig. 1, it is shown that the structure of LUT6 can be implemented using the output bit O6, or two LUT5s, which are used in the existing method and will require more logic size and power consumption. Thus, the proposed method will re-modify the LUT6 architecture using one 5-bit combinational function with one multiplexer, which will require significantly less logic size and power consumption and the architecture of the proposed LUT6 is shown in Fig. 2. A LUT6 2 with an INIT value of 0000000000000002(hex) for example, specifies that it willcreate the outputs O5 = 1 and O6 = 0 when given the combined input 100001. The execution of combinational functions is notthe only thing that is done [8].

$$S_i = C_i \oplus P_i \quad (1)$$

$$C_{i+1} = G_i + C_i \times P_i \quad (2)$$

In (1), S_i = output sum of the i^{th} block, P_i= propagation of carry from C_i to C_{i+1}. G_i = carry generate. (1) and (2) define the carry-lookahead adder, which is implemented by the carry chain, which uses the generate carry signal (O5) and the propagate signal (O6). The external bypass signals AX – DX may also supply the generate carry signals for the carry chain.

III. APPROXIMATE UNSIGNED MULTIPLIER ARCHITECTURE

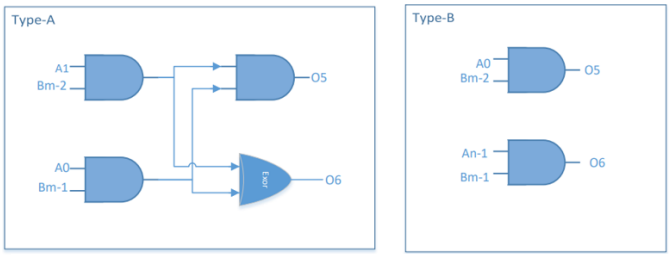
All partial products in a signed multiplication must be correctly sign-extended in comparison with unsigned multiplication for the accurate product. Because the sign informationis encoded in the created partial products, there is no longera requirement to compute and communicate sign-extension bits. Specifically, we have employed the LUT6 for calculatingthe requisite partial products (PP) by executing the AND

operation between every bit of the multiplicand and multiplier. Our automated system, on the other hand, groups every two successive partial products to maximize the usage of LUTs. There are four groups in total, each of which comprises the second partial product that has been pushed to the left by a single position concerning the first PP [9]. In addition, the PPs in each group is calculated and collectively added in a single step, saving time and effort. Although there are two PP terms per group, for example in the first group there are only two PP terms, which are A0B0 and AN1B1, that are not combined with any other PP term in their corresponding group. Furthermore, because of the restricted amount of input/output pins available on LUTs in contemporary FPGAs. So, it is not feasible to merge more than two PP terms in a single group[10].

Our technique distributes the LUT6 and accompanying carry chains to each set of PPs in this stage, which is shown in Fig. 4 by the execution blocks Type-A and Type-B. The term "block" refers to a LUT6 with an accompanying carry chain cell and an adder that may be classified as either Type-B or Type-A (CC).

The LUT of block Type-A is seen in Fig. 3(i), which depicts its operation. O5 and O6 are the output signals that are sent to the appropriate carry chain as generate carry (Gi) and propagate carry (Pi) signals in the corresponding carry chain [11]. The LUT setup for block Type-B, shown in Fig. 3(ii), employs the O5 function for the calculation of the least PP term in each row of the block. A constant '0' is used as a generate signal for the carry chain element that corresponds to block type B, and this signal is given by the external bypass signal (AX–DX) as previously detailed in Fig. 1 and Fig. 2.

To reduce PPPs (Processed Partial Products) in each group, we use compressors with a ratio of 3:1 and 2:1 in our solution. PPPs reduction might result in the creation of additional partial sums, which are then gatheredand run through the 3:1 and 2:1 compressors once again. This procedure is continued until the desired result is achieved. As seen in Fig. 4, the 8x8 multiplication architectural process generates 8 PPPs, which is equivalent to a 16x16 multiplier generating PPPs, and the process of combining and reducing these PPPs to calculate the final result.



i. Type-A LUT ii. Type-B LUT

Fig. 3. Configuration of LUTs for advanced unsigned multiplier [1]

The Type-A and Type-B LUT configurations for creating and summing the partial product bits AY BY and AXBX

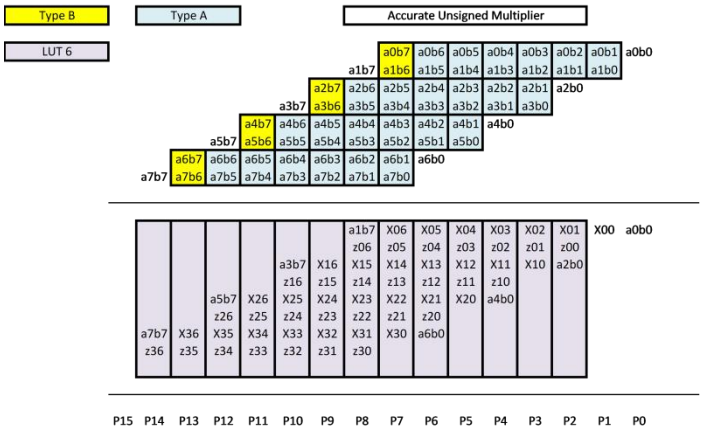


Fig. 4. The Architecture of Unsigned Multiplier using Type-A, Type-B, and LUT6

TABLE I
TYPE-A LUT CONFIGURATION [1]

Ay	By	Ax	Bx	AxBx	AyBy	AxBx + AyBy		O6	O5
						S(O6)	C(O5)		
0	0	0	0	0	0	0	0	8	0
0	0	0	1	0	0	0	0		
0	0	1	0	0	0	0	0		
0	0	1	1	1	0	1	0		
0	1	0	0	0	0	0	0	8	0
0	1	0	1	0	0	0	0		
0	1	1	0	0	0	0	0		
0	1	1	1	1	0	1	0		
1	0	0	0	0	0	0	0	8	0
1	0	0	1	0	0	0	0		
1	0	1	0	0	0	0	0		
1	0	1	1	1	0	1	0		
1	1	0	0	0	1	1	0	7	8
1	1	0	1	0	1	1	0		
1	1	1	0	0	1	1	0		
1	1	1	1	1	1	0	1		

TABLE II
TYPE-B LUT CONFIGURATION [1]

Ay	By	Ax	Bx	AxBx	AyBy	Sum (O6)	Carry (O5)	O6	O5
0	0	0	0	0	0	0	0	0	8
0	0	0	1	0	0	0	0		
0	0	1	0	0	0	0	0		
0	0	1	1	1	0	0	1		
0	1	0	0	0	0	0	0	0	8
0	1	0	1	0	0	0	0		
0	1	1	0	0	0	0	0		
0	1	1	1	1	0	0	1		
1	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0		
1	0	1	0	0	0	0	0		
1	0	1	1	1	0	0	1		
1	1	0	0	0	1	1	0	F	8
1	1	0	1	0	1	1	0		
1	1	1	0	0	1	1	0		
1	1	1	1	1	1	1	1		

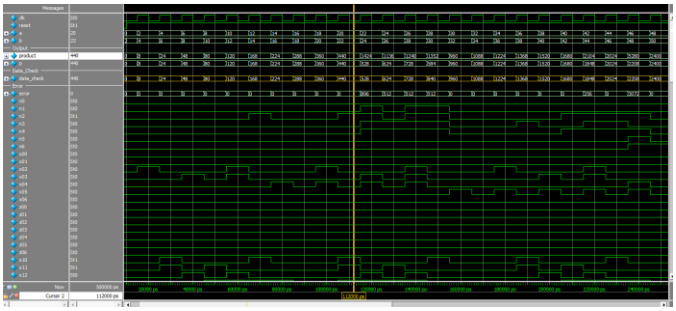


Fig. 5. Simulation results of 8x8 unsigned approximate multiplier

are defined in Tables I and II, respectively. These PPPs are organized into a single group and the final result is calculated using ternary addition. The final product bits P1–P4 are computed by combining three partial products in one step [12]. To compute additional product bits, the carry out of a given slice is sent to the following slice's carry chain.

Although the proposed unsigned 8x8 approximate multiplier was developed in Verilog HDL and synthesized in Vertex-5 FPGA, its simulated output was verified using ModelSim 6.5b. Fig. 5 shows the simulation results.

IV. APPROXIMATE SIGNED MULTIPLIER ARCHITECTURE

We show our innovative design of an accurate signed multiplier, which is based on the suggested approximate signed multiplier of Baugh-multiplication Wooley's method and the proposed approximation signed multiplier. This algorithm creates just M/2 signed PPPs for a signed multiplier with a N x M coefficient. This characteristic of our suggested solution is equivalent to the widely used radix-4 Booth's multiplication technique, which reduces the total number of PPs created by half. Moreover, Baugh-approach Wooley removes the requirement for additional extension sign bits, which makes it possible to construct the multiplier in a more resource-efficient manner. A compliment is formed by combining the final PP row with the most important word in all previousPP rows, as shown in the example [13-14]. The creation of these supplemented words necessitates a modification to our proposed design flow, which includes the addition of three additional LUT configurations.

The suggested technique for a N x M signed multiplier is shown in Fig. 7 by the LUTs and carry chain assignment step, which is based on these configurations. The final result is calculated by using the 'Re-arrangement and reduction of PPPs' phase of the proposed technique after creating all signed PPPs and before doing any further processing on them [15].

As illustrated in the preceding section, the suggested signed multiplier design makes use of ternary and binary adders for the addition of partial products formed during the calculation

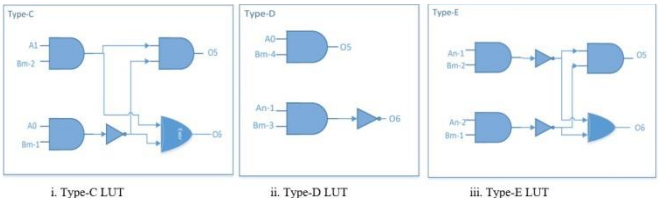


Fig. 6. Configuration of LUTs for advanced signed Multiplier [1]

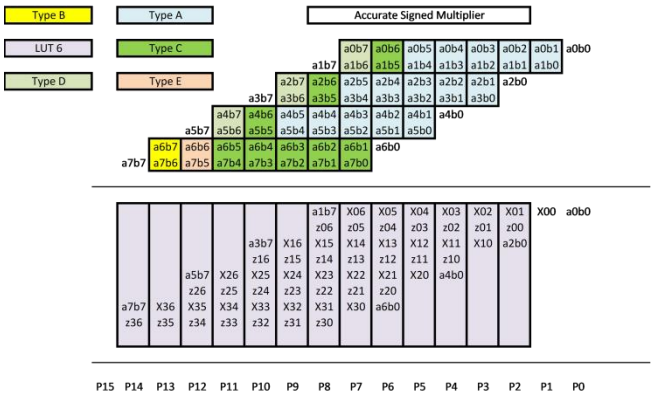


Fig. 7. The Architecture of signed Multiplier using Type-C, Type-D, Type-E, and LUT6

process [16-17]. In addition, the use of ternary adders permits area-efficient implementations in signed accurate multiplier simulation. The proposed signed 8x8 multiplier was developed in Verilog HDL and synthesized in vertex-5 FPGA, its simulated output was verified with ModelSim 6.5b. Fig. 8 shows the simulation results.

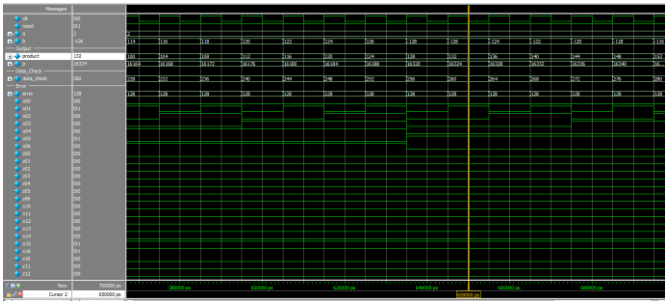


Fig. 8. Simulation results of signed 8x8 Approximate Multiplier

V. IMPLEMENTATION AND RESULTS

In the beginning, the design of some basic approximate multipliers is concentrated. Higher-order approximate multipliers are implemented by making use of sub multipliers in the following sections. It is essential, while developing a performance/area optimized elementary multiplier module for FPGAs, to take advantage of the LUT6 structure and the associated carry chains that are offered by a certain FPGA. This is because the LUT6 structure is accessible in the FPGA. The only multiplier designs that are even somewhat feasible are 8 x 8 multipliers since they are the only ones that can use all of the inputs

TABLE III
COMPARISON OF ACCURATE AND APPROXIMATE SIGNED AND UNSIGNED 8 X 8 MULTIPLEXERS

	Signed Multiplier		Unsigned Multiplier	
	Proposed	Existing	Proposed	Existing
No.of Slice LUTs	81	94	78	94
No.of Occupied Slices	21	84	22	84
No.of bonded IOBs	32	32	32	32
Delay (ns)	13.818	15.665	14.067	15.064

of a LUT6. Because of the limited applications for an 8 x 8 multiplier, we have chosen to exclude it from our selection of basic multiplier modules. This decision was reached because of the restricted usage of such a multiplier. The only possible basic design is a 4 x 4 multiplier that makes significant use of the lookup tables of current-generation FPGA devices.

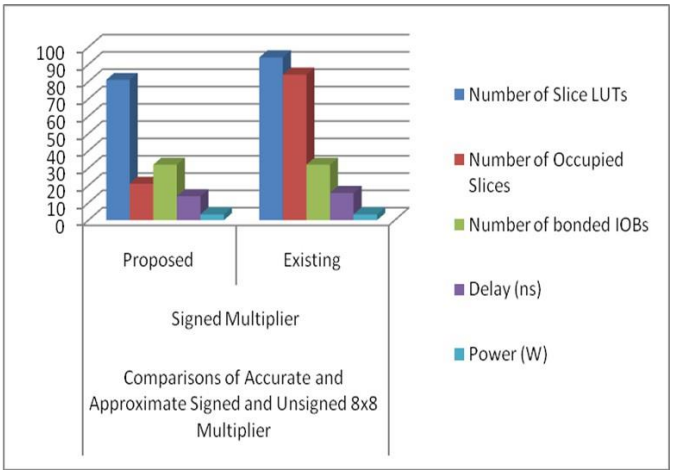


Fig. 9. Comparison Analysis of Accurate and Approximate Signed Multiplier

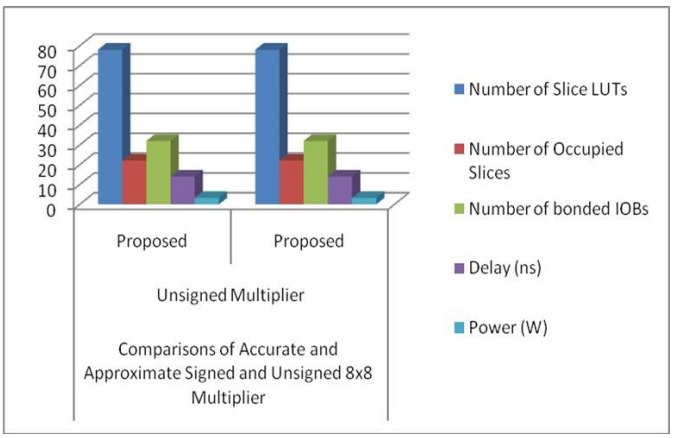


Fig. 10. Comparison Analysis of Accurate and Approximate unsigned Multiplier

This is the only design that can be implemented. In the present investigation, the signed and unsigned 8 x 8 multiplier is utilized as the primary constituent in the construction of higher-order approximation multipliers as the basic building block. The method that is presented in this study should be utilized instead of a multiple LUT5 in LUT6 architecture for building signed and unsigned multipliers for 8-bit setups. This would result in multipliers that are both more accurate and approximately correct [18]. All of the metrics were measured and compared in terms of area, latency, and power throughout the development, which was carried out in the software provided by Xilinx and Verilog HDL. Analyze the differences in the accuracy and approximations of signed and unsigned 8x8 multipliers, which are described in Table III, and compared the results. The comparisons and analyses of accuracy and approximation in the signed multiplier chart displayed in Fig. 9 and the unsigned 8x8 multiplier chart shown in Fig. 10 are shown.

VI. CONCLUSION

As part of this research, we provide area-optimized, high-performance soft-core exact and approximation multipliers that make use of the LUT6 tables and supporting sum and carry chains seen in current FPGAs. It is recommended that LUT6 with multiplexers be used instead of numerous LUT5s to create accurate and approximation signed and unsigned multipliers for 8-bit configurations. In comparison to Xilinx's area-efficient multiplier IP, we propose accurate signed and unsigned multipliers that may lower the total number of LUTs utilized by as much as 62 percent, depending on the application. For a M x N accurate multiplier, this approach produces just N/2 partial products in parallel, which is a significant reduction in complexity. When it comes to calculating the final result, the ternary and binary adders come in handy, respectively. Using our proposed approximate 4x2 and 4x4 multipliers, we have developed higher-order approximate multipliers using the modified technique, which we have compared to previous work. As a consequence, we have unveiled the design of an efficient and high-performance adder for the output of sub multipliers, which was implemented in software. In this work, for a signed multiplier, a notable decrease in area and delay by 42.69% and 11.79% was observed and for an unsigned multiplier, 43.8% and 6.61% reduction were attained when compared with that of the architecture of existing LUT6.

REFERENCES

[1] Salim Ullah, Semeen Rehman, Muhammad Shafique and Akash Kumar, May 2021, IEEE, High Performance Accurate and Approximate Multipliers for FPGA based Hardware Accelerators.

[2] S. Karthick, C. Kamalanathan, P. Sunita, S. Ananthakumaran, and E. Prabhu, "High speed energy efficient multiplier for signal processing", International Journal of Engineering Systems Modelling and Simulation, Vol.12, No.4, pp.221-229, 2021.

[3] Antonio Giuseppe Maria Strollo , Senior Member, IEEE, Ettore Napoli , Senior Member, IEEE, Davide De Caro , Senior Member, IEEE, Nicola Petra, Member, IEEE, and Gennaro Di Meo, Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers, IEEE, March 2020.

[4] C. Mahitha, S. C. S. Ayyar, S. D. B, A. Othayoth and R. S R, "A Low Power Signed Redundant Binary Vedic Multiplier," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 76-81, doi: 10.1109/ICOEI51242.2021.9453032.

[5] Honglan Jiang , Student Member, IEEE, Cong Liu, Fabrizio Lombardi , Fellow, IEEE, and Jie Han, Low-Power Approximate Unsigned Multipliers With Configurable Error Recovery, IEEE, January 2018.

[6] Mohammad Khaleqi Qaleh, Mohammad Ahmadinejad, Mohammad Hossein Moaiyeri, Ultraefficient imprecise multipliers based on innovative 4:2 approximate compressors, July 2020, Wiley.

[7] Karri Manikantta Reddy, M.H. Vasantha, Y.B. Nithin Kumar, Devesh Dwivedi, Design and analysis of multiplier using approximate 4-2 compressor, Elsevier, 2019, MAY
[8] Seyyed Ashkan Ebrahimi, Peiman Keshavarzian, Saeid Sorouri, Mahyar Shahsavari, Low Power CNTFET based Ternary Full Adder Cell for Nanoelectronics, International Journal of Soft Computing and Engineering (IJSCE), May 2012.
[9] Prabhu E., Mangalam, H. Gokul, P.R. "A Delay Efficient Vedic Multiplier", Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, vol. 89, no.2, pp. 257-268, 2019.
[10] N. Brunie, F. de Dinechin, M. Istoan, G. Sergent, K. Illyes and B. Popa, "Arithmetic core generation using bit heaps," 2013 23rd International Conference on Field programmable Logic and Applications, Porto, 2013, pp. 1-8.
[11] J. Beuchat and J. Muller, "Automatic Generation of Modular Multipliers for FPGA Applications," in IEEE Transactions on Computers, vol. 57, no. 12, pp. 1600-1613, Dec. 2008.
[12] Ahmet Kakacak, Aydin Emre Guzel, Ozan Cihangir, Sezer Gren, and H. Fatih Ugurdag. 2017. "Fast Multiplier Generator for FPGAs with LUT based Partial Product Generation and Column/row Compression,". in Integr. VLSI J. 57, C 2017, 147-157.
[13] M. Kumm, J. Kappauf, M. Istoan and P. Zipf, "Resource Optimal Design of Large Multipliers for FPGAs," 2017 IEEE 24th Symposium on Computer Arithmetic (ARITH), London, 2017, pp. 131-138.
[14] P. K. Somayajulu and Ramesh. S R, "Area and Power Efficient 64-Bit Booth Multiplier," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 721-724. doi: 10.1109/ICACCS48705.2020.9074305
[15] E. G. Walters, "Array Multipliers for High Throughput in Xilinx FPGAs with 6-Input LUTs" in Computers, vol. 5, no. 4, 2016.
[16] M. Kumm, S. Abbas and P. Zipf, "An Efficient Softcore Multiplier Architecture for Xilinx FPGAs," 2015 IEEE 22nd Symposium on Computer Arithmetic, Lyon, 2015, pp. 18-25.
[17] H. Parandeh-Afshar and P. Ienne, "Measuring and Reducing the Performance Gap between Embedded and Soft Multipliers on FPGAs," 2011 21st International Conference on Field Programmable Logic and Applications, Chania, 2011, pp. 225-231.
[18] H. Parandeh-Afshar, P. Brisk and P. Ienne, "Exploiting fast carry-chains of FPGAs for designing compressor trees," 2009 International Conference on Field Programmable Logic and Applications, Prague, 2009, pp. 242- 249.