International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Area and Power Efficient Multipliers Using Approximate Compressors and Full Adders for DSP Applications

Shaheen[1], Dr. Anindya Jana[2]
M. Tech (VLSISD)[1], Associate Professor[2]
[1&2]Department. of ECE, JBIET, Hyderabad Telangana, India

Article Info
Volume 9, Issue 5
Page Number : 546-554

Publication Issue
September-October-2022

Article History
Accepted : 10 Oct 2022
Published : 28 Oct 2022

ABSTRACT

In this project, a new approximate full adder is proposed and applied in the reduction stages of multiplier. Using the current approximate 4-2 compressors as well as proposed approximate full adder designs are used for developing an 8-bit multiplier. For applications that are error robust, approximate computing can lessen design complexity while boosting performance and power efficiency. We can learn a lot from marginally inaccurate outputs in the majority of multimedia apps. Consequently, we are not required to create precise outcomes. In order to benefit from the relaxation of numerical exactness, this brief discusses a new technique to gate level logic modification for full adder approximation. The sum term of the conventional full adder is altered to reduce an area complexity. The effectiveness of the proposed method is synthesized and simulated using Xilinx Vivado.

Keywords :- Approximate full adder, inaccurate multiplier, and approximate compressor.

I. INTRODUCTION

An emerging strategy in digital design is approximation computing, which aims to provide considerable performance increase in design parameters. For embedded and mobile systems, which are defined by stringent energy and speed limits, this strategy is becoming more and more crucial.

Approximate computing may be used in a variety of error-tolerant applications. Examples include machine learning, data mining, and multimedia processing. Multipliers are crucial parts of microprocessors, digital signal processors, and embedded systems, with applications ranging from filtering to convolutional neural networks. Multipliers are one of the most power consuming digital blocks, but they are also distinguished by complicated logic architecture. As a result, the design of approximate multipliers has drawn significant study attention recently.

Partially producing products, partially reducing produced products, and carrying out carry-propagate addition are the three fundamental building components of a multiplier. Any of these building blocks may include approximations. This approach avoids the formation of partial products and, by using the proper correction functions, minimizes truncation error.

Energy-efficiency has become the paramount concern in design of computing systems. At the same time, as the computing systems become increasingly embedded and mobile, computational tasks include a growing set of applications that involve media processing (audio, video, graphics, and image), recognition, and data mining.

A common characteristic of the above class of applications is that often a perfect result is not necessary and an approximate or less-than-optimal result is sufficient. It is a familiar feature of image processing. Such applications are imprecision-tolerant. The primary purpose of this paper is to review the recent developments in the area of approximate computing (AC). The common underlying thread in these disparate efforts is the search for solutions that allow computing systems to trade energy for quality of the computed result. In this paper we focus on the solutions that involve rethinking of how hardware needs to be designed.

A developing method in digital design called approximation computing relaxes the need for precise calculation in order to greatly improve performance in terms of power, speed, and area.

A new development in digital design is approximate computing, which compromises accuracy for increased performance in terms of speed and power. In the paper they proposed a novel approximate compressors and 8 bit and 16 bit multiplier designs to evaluate the performance of the proposed compressors.

This article's goal is to present a thorough analysis and a comparative assessment of recently created approximate arithmetic circuits. Applications of these circuits in the field of image processing and deep neural networks show that the circuits with lower error rates or error biases perform better in straightforward calculations like the sum of products. A higher approximation can be accepted in multipliers than in adders since such complex computations are more susceptible to addition errors than multiplication errors. Along with benefits for performance and power consumption, the usage of approximate arithmetic circuits can enhance the quality of deep learning and image processing.

II. EARLIER WORK

The previous or existing method involves Dadda multiplier using exact multipliers. These exact multipliers uses 5 inputs and produces three outputs.

The major emphasis of the field's research was the construction of parallel digital multiplier circuits, which were significantly less optimised than adder circuits.

Dadda’s work reduces the amount of additions and optimizes their arrangement to shorten propagation latency. Comparing this solution to earlier ones, especially the Wallace one that comes just before it, the Wallace parallel multiplier method dramatically reduces the count of logic gates used.

In compared to the techniques that were in use at the time, the idea behind the Dadda multiplier methodology, which includes delaying and spreading the carry propagation. The two recognized designs for such a fundamental arithmetic circuit type are the Wallace design and the Dadda approach for parallel multipliers, formerly known as the Dadda tree.

Since then, one of the two often used parallel multiplier systems taught in university courses on computer arithmetic is the Wallace trees, commonly referred to as Dadda trees. Compressing the partial products more quickly is a function of the 4-2 compressors used.

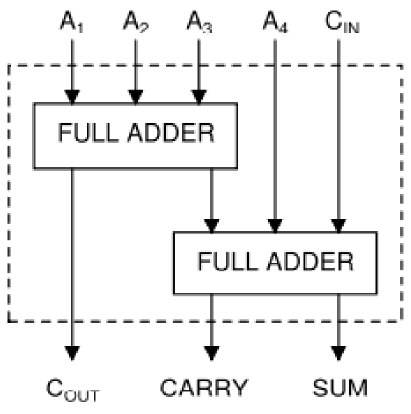


Fig.1: Exact compressor

There are five inputs overall, two linked full adders, and three total outputs. A1, A2, A3, A4, and CIN are the inputs of the exact 4:2 compressor. Its outputs are: COUT, CARRY, and SUM. It is stated that COUT, CARRY, and SUM

$$C_{OUT} = A_3(A_1 \oplus A_2) + A_1(\overline{A_1 \oplus A_2}) \qquad (1)$$

$$CARRY = C_{IN}(A_1 \oplus A_2 \oplus A_3 \oplus A_4) + A_4(\overline{A_1 \oplus A_2 \oplus A_3 \oplus A_4}) \qquad (2)$$

$$SUM = C_{IN} \oplus A_1 \oplus A_2 \oplus A_3 \oplus A_4 \qquad (3)$$

Figure 2 illustrates a compressor chain. The lower significant bits of the previous 4: 2 compressor's input carry are represented by CIN. The order 1 outputs with more importance than the input CIN are CARRY and COUT.S

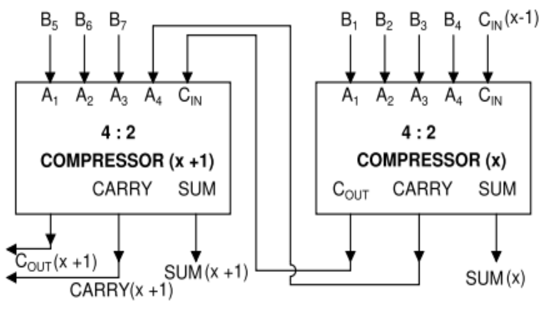


Fig.2: compressor chain

A_1	A_2	A_3	A_4	C_{IN}	C_{OUT}	$CARRY$	SUM
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	1	0	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

Fig.3: Truth table of exact compressor

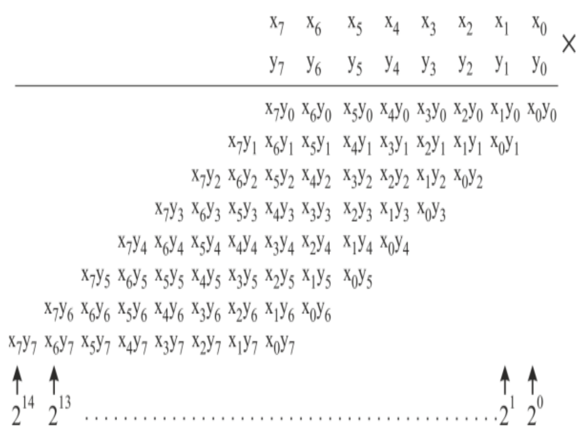


Fig.4: Partial product matrix of an 8 × 8 bit Multiplier.

The partial products of this column's arithmetic sum, S, are denoted as follows:

$$S = \sum \{p_0, p_1, p_2, \ldots p_{j-1}\}$$

The arithmetic total in (2) is computed by a compressor, who then encodes the findings in binary form. The most popular compressor uses a full-adder with three inputs and two outputs: sum and carry.

A 4/2 compressor have five inputs, one of which is a carry from a column to the right, and encodes the result on three outputs: sum, carry1, and carry2.

The total, carry1, carry2, carry3, and two carries from a column to the right are the four outputs that the 5/2 compressor uses to encode the result. Two of the seven inputs to the 5/2 compressor are carry (having double weight).

The ability to do arithmetic operations requires multipliers in both microprocessors and digital signal processors. The performance characteristics of either DSPs or microprocessors would be improved by implementing the efficient and effective multiplication algorithm. The basic building elements of digital systems are multipliers.

Multipliers have an impact on both the computational efficiency and power used by the digital system. Therefore, it is essential for a digital system to have high speed multipliers with low power dissipation. As a result, it can make digital systems more effective.

compression of columns Due to their rapid computation, multipliers have grown in favor.

Column compression multipliers by Wallace and Dadda are widely known. In 1964, Chris Wallace, an Australian computer scientist, put up the idea for the Wallace Multiplier.

Italian computer engineer Luigi Dadda developed the Wallace Multiplier before suggesting an improvement with the Dadda Multiplier. Dadda and Wallace multipliers both rely on reduction to operate. A [3, 2] counter and a [2, 2] counter are used to compress the columns to reduce their size. Wallace and Dadda Multipliers use the same three phases in their respective processes.

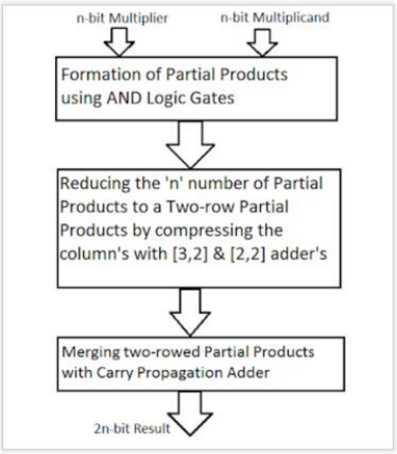

n-bit Multiplier
n-bit Multiplicand
Formation of Partial Products using AND Logic Gates
Reducing the 'n' number of Partial Products to a Two-row Partial Products by compressing the column's with [3,2] & [2,2] adder's
Merging two-rowed Partial Products with Carry Propagation Adder
2n-bit Result

Fig. 5 Flowchart of Multiplication process

Dadda proposed a reduction method that, in the fewest number of reduction stages, yields the reduced two-rowed Partial products. Dadda managed to do this by placing the [3, 2] and [2, 2] counters in the best possible positions on the maximum Critical path.

For an N-bit multiplier and N-bit multiplicand, a partial product of N by N is generated. They are organised into what resembles a grid. Dadda reduced this Matrix height to a two-row matrix by employing a number of reduction methods.

Algorithm:

1. Assume that the ultimate height of the two-row matrix, d1, is 2, and that subsequent matrix heights, dj+1 = 1.5 * dj, where j = 1, 2, 3, 4, etc., are obtained from dj depending on d1. In this matrix height, the fraction needs to be rounded to the nearest whole number.

In this manner, the matrix heights will be 2, 3, 4, 6, 9, 13, 19 and 28. So that the resulting matrix height doesn't surpass the total height of the matrix, the greatest dj should then be determined.

2. The column compression in the first reduction step should be made with the help of [3, 2] and [2, 2] counters in a way that ensures the height of the reduced matrix produced does not exceed dj.

3. The carry must be transferred to the next column and the total should pass to the same column in the subsequent reduction step during the compression.

4. Repeat steps 2 and 3 until a final reduced matrix with two rows is created.

Three approximately 4-to-2 compressors are shown in this study, together with an ECM (error-correcting module). The careful compensatory element of the building served as the foundation for the three 4-2 compressors. An array of approximation compressors' combined error performance is taken into account rather than just one approximate compressor's performance alone. We cleverly reduce the compressor's outputs—which ordinarily have four—to just one.

In this part, three approximately 4-to-2 compressors (UCAC1, UCAC2, and UCAC3) that are currently used in the technique are recommended. The inaccurate compensation in this instance is subsequently corrected using the ECM, which is also utilised to recognise input patterns with a high degree of probability. 8-bit multipliers based on the Dadda tree are also included in the proposed designs. Compressing incomplete items might be compared to making several additions. Additionally, the ability of the positive and negative words to cancel one another out is a crucial aspect of addition. The ED is produced by each approximation compressor in multipliers.

A chain of compressors known as a compressor chain is created by a series of cascading approximation compressors (CC).

Each compressor creates its own ED and takes up a binary bit in a CC. As a result, it is possible to think of the ED of CC as a huge binary number. To balance out the negative ED of CC, a positive ED must be introduced to the computing circuit in cases when the approximation blocks produce a negative ED. The best case scenario takes place when this block has a correction bit added to it and the compressor's ED value is -1. A this scenario an array of -1 EDs in the LSB can be efficiently balanced by a +1 in the subsequent bit of the CC's most important bit (MSB).

For the three compressors that have been discussed, TABLE I shows the truth tables. The description of UCAC1's logical purpose is provided in (6).

$$sum = y_1y_2 + (y_1 + y_2)(y_3 + y_4) + y_3y_4 \qquad (6)$$

Based on how many logic 1s there are in each of the 16 input patterns, each of the five situations is divided into 16 input patterns. Additionally, they are designated as examples zero, one, two, three, and four, respectively. A and B are symbols.

One may estimate that 1/4 of the partial products will be logic 1 since the AND gate array was employed to build them.

Notably, the bulk of the total instances in TABLE I are represented by cases 0, 1, and 20. The ED of UCAC1 is 0 or -1 in these three situations. The ED is bigger than -1 in examples three and four. The chance of these two scenarios, however, is just 13/256. By omitting the first and last terms in (6) and applying the expression of (12), we get second design, UCAC2.

$$sum = (y_1 + y_2)(y_2 + y_3) \qquad (12)$$

y_1	y_2	y_3	y_4	*UCAC1*		*UCAC2*		*UCAC3*		*pro*
				sum	*ED*	*sum*	*ED*	*sum*	*ED*	
0	0	0	0	0	0	0	0	0	0	81/256
0	0	0	1	0	-1	0	-1	1	0	27/256
0	0	1	0	0	-1	0	-1	0	-1	27/256
0	0	1	1	1	-1	0	-2	1	-1	9/256
0	1	0	0	0	-1	0	-1	1	0	27/256
0	1	0	1	1	-1	1	-1	1	-1	9/256
0	1	1	0	1	-1	1	-1	1	-1	9/256
0	1	1	1	1	-2	1	-2	1	-2	3/256
1	0	0	0	0	-1	0	-1	0	-1	27/256
1	0	0	1	1	-1	1	-1	1	-1	9/256
1	0	1	0	1	-1	1	-1	0	-2	9/256
1	0	1	1	1	-2	1	-2	1	-2	3/256
1	1	0	0	1	-1	0	-2	1	-1	9/256
1	1	0	1	1	-2	1	-2	1	-2	3/256
1	1	1	0	1	-2	1	-2	1	-2	3/256
1	1	1	1	1	-3	1	-3	1	-3	1/256

Fig. 6: THE TRUTH TABLE OF THE PROPOSED COMPRESSORS

One more riskier way to get the total of the UCAC3 is to change two words in the UCAC1's K-map from '0' to '1', which only utilises an OR gate.

$$sum = y_2 + y_4 \qquad (13)$$

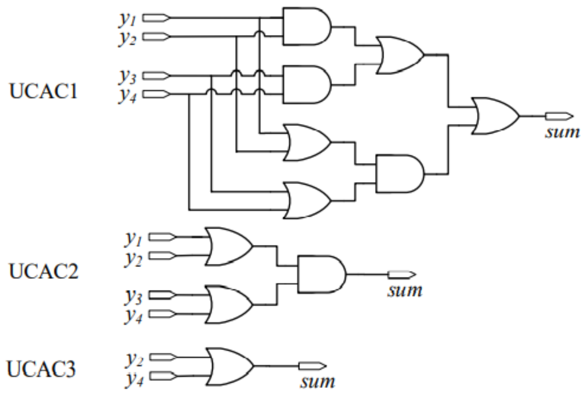
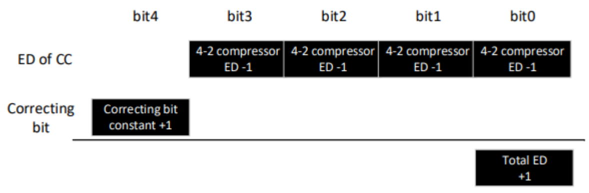


Fig. 7: (a) UCAC1 (b) UCAC2 (c) UCAC3

These suggested designs, notably the UCAC2 and UCAC3, have incredibly straightforward architectures, as seen in the Fig. above. But these compressors result in a large ED when there are more than two logic 1 inputs. They all also maintain high ERs.

In order to produce a positive error, the suggested compressors are incorporated into the multipliers, as seen in the picture below. This is done by adding a constant logic '1' to the next bit of the CC's MSB.



$$\Pr(0o) = \Pr\left(a_i \cdot b_j\right)^4 = 81/256 \qquad (7)$$

The 0 scenario has a non-negligible probability of 81/256, according to Eqn. (7), which also shows this. The suggested compressors produce the right output, nevertheless, when the 0o situation occurs. In this scenario, the correcting bit's introduction of a positive error is not countered by any negative error. In addition, the correcting bit above has a significant binary bit weight, as seen in Fig., which results in a significant total ED.

In order to change the correcting bit from logic 1 to 0 in this instance, an ECM is supplied to identify this input pattern. In spite of the fact that they cover a bigger proportion of occurrences, the compressors cause a negative error in the other circumstances, which may be sufficient to balance the correcting non-zero bit. The approximation compressor at the MSB of the CC has the same inputs that the ECM does. Only when the 0o situation occurs in the circuit's inputs does it produce logic 0.

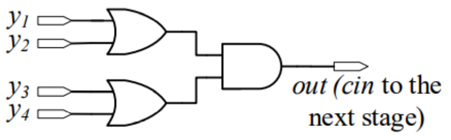


Fig. 8: Carry generation

III. PROPOSED WORK

The proposed approximate full adder's proposed approximate compressors, existing approximate compressors, and ECM's designs are used to design and assess four 8-bit multipliers, which are intended to streamline and accelerate the compression process. MUL1 stands for UCAC1 multiplier with constant correcting bit; MUL2 for UCAC1 multiplier and ECM; MUL3 for UCAC2 multiplier and ECM; and MUL4 for UCAC2 multiplier and ECM. UCAC3 multiplier and ECM; Although approximation compressors may replace any classic 4-2 compressors, the error performance suffers [5]. As a result, for N-1 less important bits in the suggested multipliers, approximate compressors are employed. The figure below shows the compression tree as well.

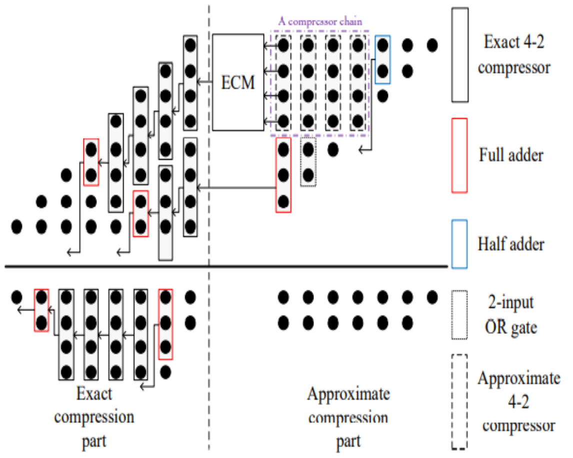


Fig. 9: proposed multiplier

As seen in this picture, the rectangles with solid or dashed lines in Figure 5 stand in for various digital circuits. The output of the precise 4-2 compressor is routed into the cin signal of the nearby ECM. The constant logic 1 in MUL1 takes thess role of the ECM. By compressing the partial products using an OR gate, which has the benefit of only requiring one compression stage to be set up in the approximation portion, the high compression ratio is effectively used. Only two numbers are added using half adder. The full adder was created next to tackle this issue. A, B, and carry C are all 1-bit binary values which gets added using the full adder. The hardware requirement in terms of full adder (FA) and the length of final adder (FAL) for different size of array multipliers is obtained in the manner given in below fig

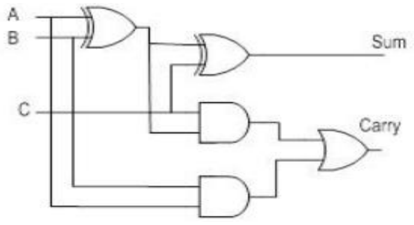


Fig. 10: Full adder

Three inputs are added by a full adder, which generates two outputs. The first two inputs are A and B, while the third input is a carry that is designated as C. The suffix SUM designates the output that is considered normal, whereas the suffix CARRY designates the output carry.

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
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. 11 Truth Table of full adder

In the above table,

- The literals A and B are the two input variables. Two important bits will be added, which are represented by these variables.
- The carry is represented by the third input, Cin. The carry bit is retrieved from the most recent lower significant position.
- The 'Sum' and 'Carry' are the output variables that define the output values.
- The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variables.

In our approximation, one XOR gate, two AND gates and one OR gates are replaced with single OR gate and AND gate for carry calculation. Sum term is calculated by inverting carry term by using an inverter. This results in error in the four cases out of sixteen cases both in sum and carry terms. This provides more simplification, while maintaining the difference between original and approximate value as similar one for more cases. The truth table of approximate full-adder can be observed from Table I.

Inputs			Accurate outputs		Approximated outputs		
A	B	C	Sum	Carry	Sum 1	Carry 1	ED
0	0	0	0	0	1	0	1
0	0	1	1	0	0	1	2
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	0
1	1	1	1	1	0	1	1

Fig. 12: Truth table for both exact and approximate full adder

Proposed approximate full adder:

As shown in Figure below One AND gate, one OR gate, and one inverter gate are used to produce it. The Boolean expressions are

$$\text{Carry} = (b \,\&\, c) + a \quad (1)$$

$$\text{Sum} = \tilde{}\text{Carry} \quad (2)$$

In this case, there is one error in Carry and 3 errors in Sum with total error distance(ED) of 3 as shown in Table I.

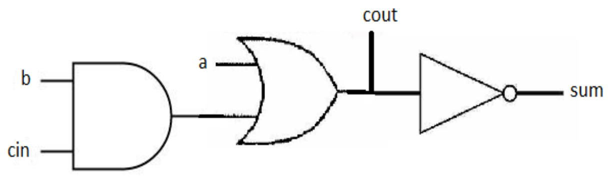


Fig. 13: approximate full adder

Based on tradeoff variables including latency, area, and power usage, the performance is compared. For low area, low power applications, the suggested approximation adder in this article can be employed. The proposed adder results in reduction of area and power cost metrics in comparison with conventional full adder.

IV. EXPERIMENTAL RESULTS

After synthesizing the Verilog code for the proposed work we can get the schematics which are RTL schematic and technology systematic shown in the fig. 14&15 respectively.

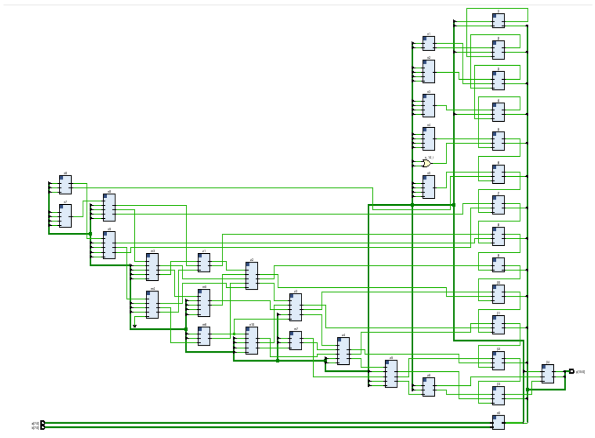


Fig.14: RTL schematic

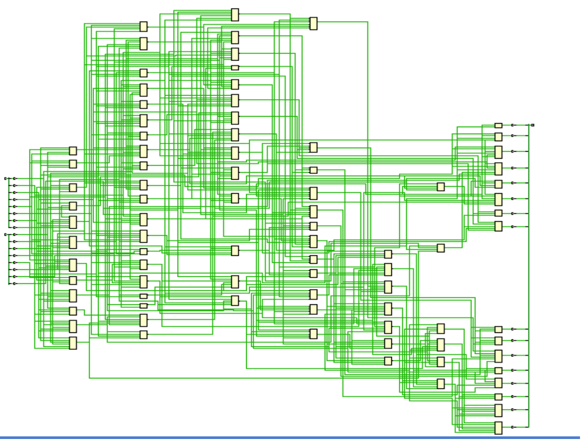


Fig.15: Technology schematic

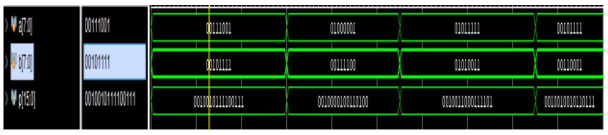


Fig.16: Simulation results

Figure 16 shows the simulation results for the proposed.

Evaluation report:

	Area (LUT's)	Delay (ns)	Power(W)
Existing design 1 [13]	79	10.684	12.953
Proposed design 1	60	9.675	12.097
Existing design 2 [13]	70	9.675	12.712
Proposed design 2	63	9.689	11.613
Existing design 3 [13]	69	9.681	12.326
Proposed design 3	64	9.779	11.924
Existing design 4 [13]	69	10.450	11.267
Proposed design 4	64	9.786	11.353

From the evaluation table it can be clearly seen that all of the design parameters are enhanced for the proposed designs when compared with the existing design.

V. CONCLUSION

In this project, we have proposed an approximate full adder to be utilized in the multiplier to reduce the area and optimize delay and power. The existing 3 approximate compressors are also considered and based on these all we have implemented 4 different types of multipliers for 8 bit. According to the simulation analysis, the proposed designs significantly increase the performance of multipliers. In addition, with the exception of error in output, the proposed multipliers maintain a tolerable error with increased performance. Additionally, they function effectively when image multiplication is used. As the findings reveal, the multipliers integrated with our approximate full adder good tradeoff between parameters and error.

VI. REFERENCES

[1]. C.–H. Chang, J. Gu, and M. Zhang, Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits, IEEE Trans. Circuits and Syst. I: Reg. Papers, vol. 51, no. 10, pp. 1985-1997, Oct. 2004.

[2]. A. Wang, B. H. Calhoun, and A. P. Chandrakasan, Sub-Threshold Design for Ultra Low-Power Systems, vol. 95. New York, NY, USA: Springer, 2006.

[3]. A. Momeni, J. Han, P. Montuschi, and F. Lombardi. Design and Analysis of Approximate

Compressors for Multiplication, IEEE Trans. Comput., vol. 64, no. 4, pp. 984-994, Apr. 2015.

[4]. Q. Xu, T. Mytkowicz, and N. S. Kim, Approximate Computing: A Survey, IEEE Design & Test, vol. 33, no. 1, pp. 8-22, Feb. 2016.

[5]. L. Qian, C. Wang, W. Liu, F. Lombardi, and J. Han, Design and evaluation of an approximate Wallace-Booth multiplier, 2016 IEEE Int. Symp. Circuits and Syst. (ISCAS), Montreal, QC, 2016, pp. 1974-1977.

[6]. O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.25, no. 4, pp. 1352-1361, Apr. 2017.

[7]. S. Venkatachalam and S.-B. Ko. Design of Power and Area Efficient Approximate Multipliers, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1782-1786, May 2017.

[8]. D. Esposito, A. G. M. Strollo, E. Napoli, D. De. Caro, and N. Petra. Approximate Multipliers Based on New Approximate Compressors, IEEE Trans. Circuits and Syst. I: Reg. Papers, vol. 65, no. 12, pp. 4169- 4182, Dec. 2018.

[9]. V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 3, pp. 421– 430, Mar. 2018.

[10]. X. Yi, H. Pei, Z. Zhang, H.Zhou, and Y. He. Design of an EnergyEfficient Approximate Compressor for Error-Resilient Multiplications, 2019 IEEE Int. Symp. Circuits and Syst. (ISCAS). Sapporo, Japan, 2019, pp. 1-5.

[11]. H. Pei, X. Yi, H. Zhou and Y. He, Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic, in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 1, pp. 461-465, Jan. 2021, doi: 10.1109/TCSII.2020.3004929.

Cite this article as :

Shaheen, Dr. Anindya Jana, "Area and Power Efficient Multipliers Using Approximate Compressors and Full Adders for DSP Applications", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 5, pp. 546-554, September-October 2022.
Journal URL : https://ijsrst.com/IJSRST229594