ISSN 2277-2685
IJESR/Jan-Mar. 2025/ Vol-15/Issue-1s/594-603
**Ganugula Harika *et. al.*, / International Journal of Engineering & Science Research**

# Area And Power Efficient VLSI Architecture Of Approximate Multiplier Using Majority Logic

**Ganugula Harika[1], Dr.M.Rajan Babu[2]**

[1]Research scholar, Lendi institute of engineering and technology, Vizag-Viziangaram Road-NH-43, Denkada, Jonnada, Andhra Pradesh 535005

[2]Professor, Lendi institute of engineering and technology, Vizag-Viziangaram Road-NH-43, Denkada, Jonnada, Andhra Pradesh 535005

***ABSTRACT***

*In this work, we present an approximate multiplier architecture that leverages a 6:3 compressor using majority gates, coupled with a Carry Look-Ahead Adder (CLA), and compare its performance to a existing multiplier using a Ripple Carry Adder (RCA) with the same 6:3 compressor. The primary aim of this study is to evaluate the efficiency, speed, and power consumption of the approximate multiplier designs. We demonstrate that the multiplier using a CLA achieves superior results in terms of reduced delay and enhanced overall performance compared to the RCA-based design. The CLA's parallel carry propagation mechanism significantly minimizes the critical path delay, leading to faster computations, while the use of majority gates in the 6:3 compressor ensures high-level approximate operations that can tolerate error for the sake of improved speed and reduced power consumption. Experimental results show that the CLA-based approximate multiplier outperforms the RCA-based design in both speed and power efficiency, making it an attractive solution for applications that can tolerate approximation in arithmetic computations.*

***Key words****— Approximate adder, approximate compressor, approximate computing (AC), approximate multiplier, image processing, majority logic (ML) and Carry look ahead adder.*

## 1. INTRODUCTION

The increasing integration of circuits, the traditional CMOS technologies have been gradually limited in the design of VLSI circuits. The power dissipation of computing systems is still an increasingly serious problem, despite advances in semiconductor technology and energy efficient design techniques [1].As a new computing paradigm at the nanoscale, approximate computing (AC) offers a promising solution to the VLSI industry by trading precision for reduced complexity and power consumption.AC takes advantage of the inherent error tolerance of the application to balance performance and accuracy of the circuit [2]. As a result, AC can be applied to many applications and architectures, such as data analysis, image recognition, multimedia, and signal processing [3], [4]. There have been a number of emerging nanotechnologies proposed in recent years, including quantum-dot cellular automata (QCA) [5], nanomagnets logic [6], and spin-wave devices [7]. These techniques are based on the majority logic (ML) abstraction, which differs from the traditional Boolean logic. The intrinsic energy consumption of nanotechnology is lower than that of CMOS. Also,

the ML function is more expressive than these traditional two-input Boolean logic operations. Thus, this article uses ML to implement the proposed designs for approximate circuits. Adders and multipliers are arithmetic units that are widely used in computing systems. Thus the performance of computing systems is significantly influenced by the speed and power consumption of arithmetic circuits. Although researchers have proposed a variety of designs for the approximate circuit in the transistor-based technologies [8], [9], [10], these designs are less attractive when implemented in other non-transistor or technologies that use different logic gates. As an example, the design shown in adopts a lot of XOR operations for carry generation and propagation. In this article, we propose both ML-based approximate full adders (MLAFAs) and ML-based approximate multipliers (MLAMs). We propose an approximate parallel 6:3 compressor and show how it can be used in combination with the Wallace-based distinctively partial product reduction (PPR) circuitry to produce a simple and efficient 8 × 8 multiplier. As an alternative to the conventional 4:2 compressor, the proposed compressor can compress six partial products simultaneously with a simpler circuit structure.

**ML-BASED EXACT FULL ADDER**

Given three Boolean variables A, B, and carry input Cin, the carry output operation of an ML-based exact fulladder (MLEFA) is natively a majority gate, i.e., Cout = M(A, B,Cin). The summation function actually acts as a three-input XOR gate. Exact synthesis can be used to find optimal logic expressions based on specified logic primitives. In terms of the number of majority gates, at least three majority gates are required. The implementation proposed in reveals that MLEFA requires three ML gates and two inverters as shown in Fig. 2, where S = M(Cout, M(A, B,Cin),Cin). Another alternative realization of the summation operation is S = M(Cin, M(A, B,Cin), M(A, B, M(A, B,Cin))), in which only one inverter is required but the logic depth is increased from 2 to 3.

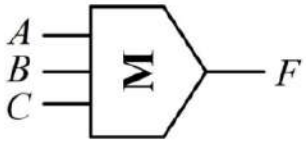


**Fig. 1.** Schematic of the majority gate

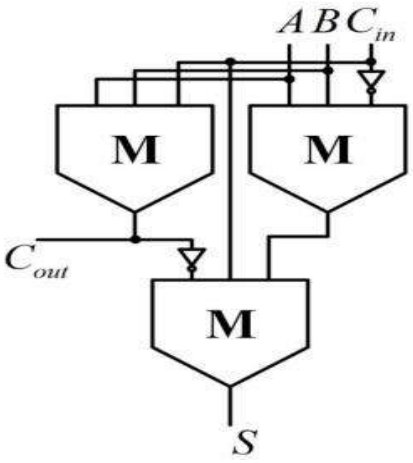


**Fig. 2.** Schematic of the exact full adder

## 2. LITERATURE SURVEY

The concept of error tolerance in approximate computing has been a central theme in the design of approximate multipliers. The key benefit of approximate multipliers lies in their ability to allow small errors in the output, which can significantly improve performance metrics like speed and power. Researchers such as **A. D. P. M. et al. (2020)** and **S.**

**V. S. S. Srinivas et al. (2017)** have studied how the magnitude of error can be controlled, particularly in the context of signal processing and machine learning, where approximate multipliers are particularly effective. Their findings suggest that slight inaccuracies in multiplication do not always negatively impact the overall performance of these applications, and in many cases, they can even be beneficial due to reduced power and faster computation.

Majority gates, when utilized in compressors, provide a simple yet effective approach to approximate arithmetic. **V. S. Neelakantan et al. (2018)** explored the use of majority gates in compressor circuits, particularly the 6:3 compressor, which has been shown to perform well in approximate multiplier designs due to its low complexity and reduced error propagation. The key advantage of majority gate-based designs lies in their ability to reduce circuit depth, making them suitable for use in low-power, high-speed applications. Additionally, majority gates offer an inherently fault-tolerant behavior, which is a desirable feature in approximate computing.

## 3. PROPOSED APPROXIMATE MULTIPLIER

A parallel 6:3 compressor and a PPR circuitry are proposed in this section that yield an efficient balance between logic implementation cost and accuracy. Then we create and use an efficient, imprecise multiplier for multiplying the images and building energy-efficient NN accelerators. A. Proposed Approximate Compressor The three steps of multiplication are: 1) partial products generation; 2) PPR; and 3) final products generation by RCA. By taking these three elements into account, PPR contributes significantly to latency, power consumption, and design complexity.

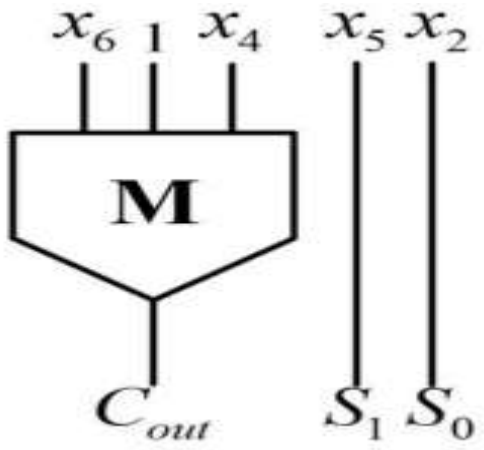


**Fig. 3.** Schematic of the approximate parallel 6:3 compressor

The proposed parallel 6:3 compressor has comparative logic implementation cost with the 4:2 compressor proposed in. The proposed compressor can be constructed with only one majority gate and no additional inverters. Despite the significantly smaller area, the MLAPC can compress six partial products simultaneously, resulting in a significant improvement in logic implementation cost.

## DESIGN OF PROPOSED PPR CIRCUITRY WITH MLAPC

The general structure of the approximate unsigned 8 × 8 Dadda multiplier based on the 6:3 compressor. Therefore, we propose a new PPR circuitry combined with the Wallace algorithm, as shown in Fig. 4 The partial products are generated using an array of majority gates with a constant “0,” which is an AND gate. In Fig. 4, each partial product bit is represented by a dot. The reduction is done using two full adders

along with 13 MLAPCs. A 7-bit CLA is then used to produce the final product. There are two possible reasons why the partial products shown by blank circles in Fig. 4 are not generated. In the proposed MLAPC, there are six inputs; however, the first and third inputs (x1 and x3) are not used, and they are not required in the production phase. Second, the outputs of the compressors are not required in the next stage. By removing the partial products indicated by the red dots in Fig. 4, additional area can be saved in the multiplier.

MLAPC does not have a symmetric input, which causes the error to be determined by the order in which the partial products are connected to the inputs. As a result, the output value may change if the inputs are permuted. Therefore, the error of the PPR tree depends on the specific connections of each partial product to each input of the approximate compressor. This was ignored in previous works. Based on a uniform and independent distribution of the inputs, we assume that all the partial products are independent of each other and their probability of being "1" (as indicated simply by a probability below) is 1/4 (since inputs "00," "01," "10," and "11" result in the outputs "0," "0," "0," and "1," respectively). As a result, there is no preferential connection between the partial products of the first stage and the approximate compressor inputs.

However, the probability of some partial products in the second stage has changed after the first stage has been reduced by MLAPCs. The blue dots in Fig. 4 represent the partial product with a probability of 7/16. Since the accuracy of MLAPC is more influenced by Cout, it is crucial to assign the partial products with different probabilities. There are three distribution cases. 1) Case 1: The probabilities of two inputs of the Cout signal generator are both 7/16. 2) Case 2: The probabilities are 7/16 and 1/4, respectively. 3) Case 3: Both the inputs have probabilities of 1/4. As a result of practical experiments, case 3 works best in the approximate multiplier. C. Image Processing Using Approximate Multipliers We apply an unsigned 8 × 8 multiplier to validate the performance of approximate multipliers for image processing.

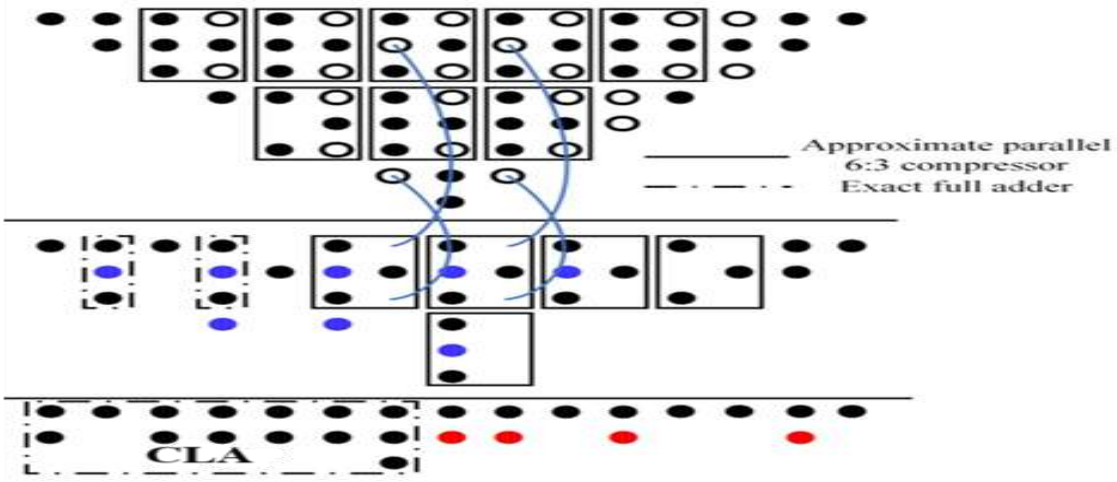


**Fig.4.** Reduction process of an unsigned 8 × 8 multiplier proposed PPR circuit

**CARRY LOOK AHEAD ADDER**

The carry look ahead adder generates carry in O (logn) time and is widely considered as the fastest adder and is widely used in the industry for high performance arithmetic circuits. In CLA, carries are computed fast by computing them in parallel at the cost of increased area. The corresponding Boolean expressions are given here to construct a carry lookahead adder. In the carry-lookahead circuit we need to generate the two signals carry propagator(P) and carry generator(G),

$$P_i = A_i \text{ xor } B_i$$

$$G_i = A_i \text{ and } B_i$$

The output sum and carry can be expressed as

$$Sum_i = P_i \text{ xor } C_i$$

$$C_{i+1} = G_i + ( P_i \cdot C_i)$$

Having these we could design the circuit. We can now write the Boolean function for the carry output of each stage and substitute for each Ci its value from the previous equations:

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 (G_0 + P_0 \cdot C_0)$$

$$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot (G_1 + P_1 (G_0 + P_0 \cdot C_0))$$

$$C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 (G_0 + P_0 \cdot C_0))$$

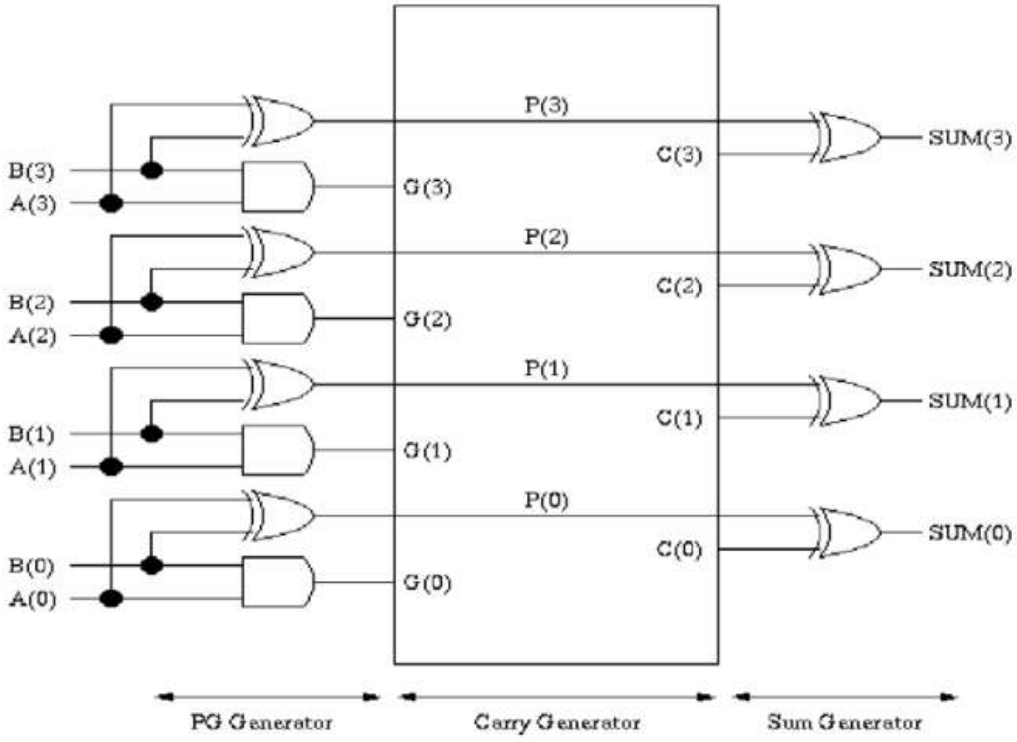


**Fig.5.** Carry look ahead adder

## 4.RESULTS

RTL SCHEMATIC: The RTL schematic is abbreviated as the register transfer level it denotes the blue print of the architecture and is used to verify the designed architecture to the ideal architecture that we are in need of development .The HDL language is used to convert the description or summery of the architecture to the working summery by use of the coding language i.e verilog ,vhdl. The RTL schematic even specifies the internal connection blocks for better analyzing. The figure represented below shows the RTL schematic diagram of the designed architecture.

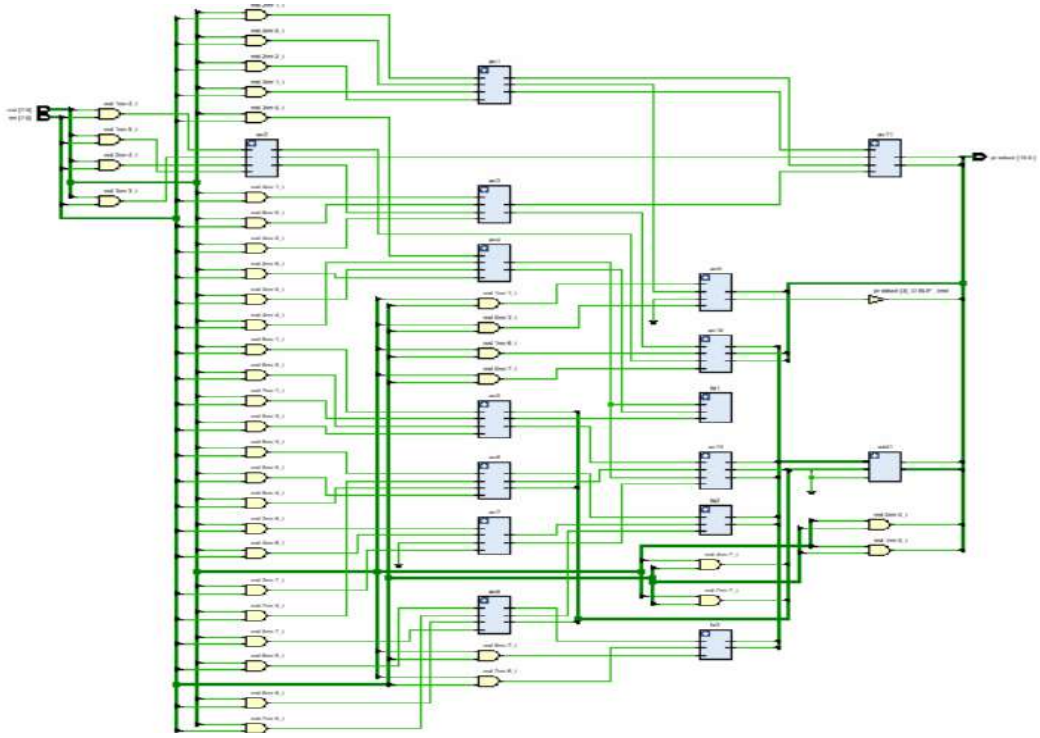

**Fig. 6.** RTL Schematic of proposed design

TECHNOLOGY SCHEMATIC:The technology schematic makes the presentation of the architecture in the LUT format ,where the LUT is consider as the parameter o area that is used in VLSI to estimate the architecture design .the LUT is consider as an squarunit the memory allocation of the code is represented in there LUT s in FPGA.

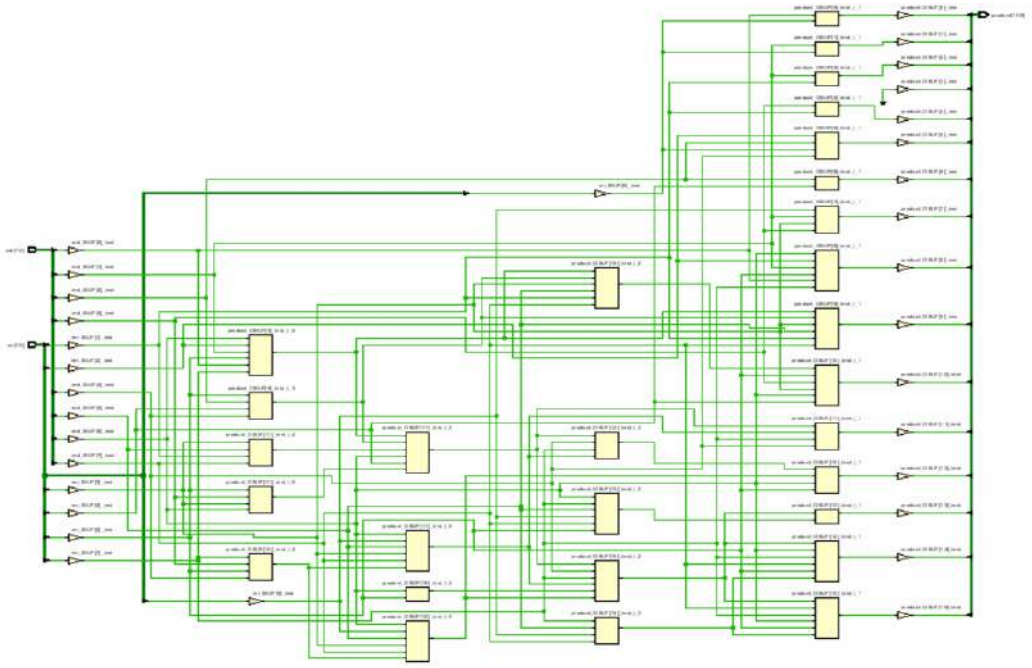

**Fig. 7.** View Technology Schematic of proposed design

SIMULATION:The simulation is the process which is termed as the final verification in respect to its working whereas the schematic is the verification of the connections and blocks. The simulation window is launched as shifting from implantation to the simulation on the home screen of the tool and the simulation window confines the output in the form of the wave forms. Here it has the flexibility of providing the different radix number systems.


**Fig.8** Simulated Waveforms of proposed design

PARAMETERS: Consider in VLSI the parameters treated are area, delay and power, based on these parameters one can judge the one architecture to other. here the consideration of delay is the

parameter is obtained by using the tool XILINX 14.7 and the HDL language is Verilog language.

| Parameter | Approximate multiplier using RCA (Existing design) | Approximate multiplier using CLA (Proposed design) |
|---|---|---|
| Number of LUTs | 26 | 23 |
| Power (mw) | 7.274 | 6.986 |

**Table. 1.** Parameter comparison

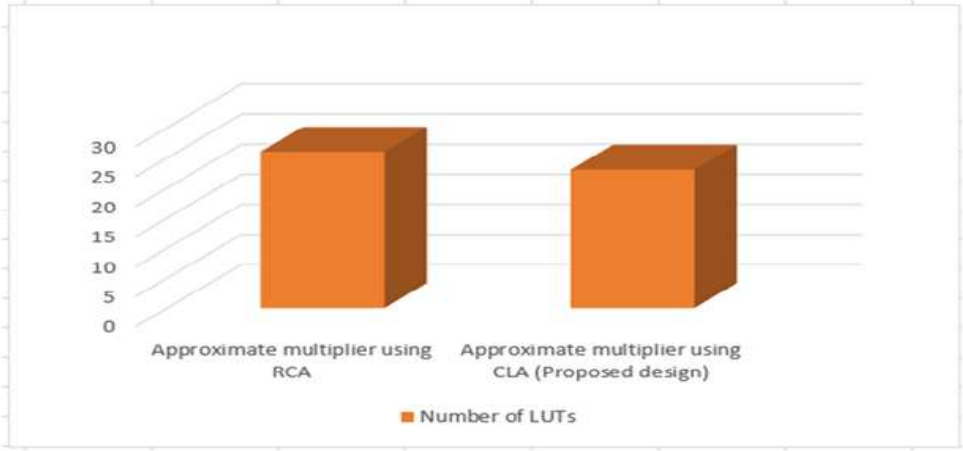


**Fig. 9.** LUT comparison Bar-graph

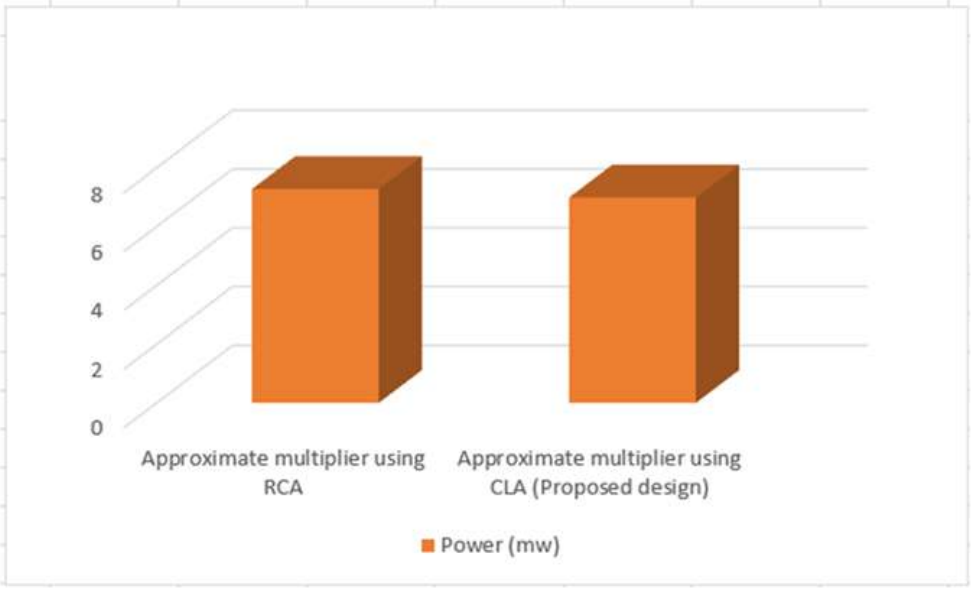


**Fig. 10.** power comparison Bar-graph

**CONCLUSION**

In conclusion, the use of approximate multipliers utilizing 6:3 majority-gate compressors, combined

with either Ripple Carry Adders (RCA) or Carry Look-Ahead Adders (CLA), offers a promising approach for enhancing the performance of computational systems where some level of error tolerance is acceptable. The key advantages of these designs include improved speed (due to reduced carry propagation delay with CLA), lower power consumption, reduced hardware complexity, and scalability for various bit-width operations.

Approximate multipliers are particularly beneficial in applications where exact precision is not critical, such as digital signal processing, machine learning, image and video processing, embedded systems, and real-time communication systems. They offer a balance between performance and error tolerance, enabling faster computations and more efficient use of resources, while still maintaining acceptable output quality.The choice between RCA and CLA-based multipliers depends on the specific performance requirements of the application. While CLA-based multipliers deliver higher speed at the cost of slightly increased complexity, RCA-based designs are simpler and still suitable for many applications with less stringent speed demands.Overall, approximate multipliers provide a cost-effective solution for various high-performance, power-constrained, and resource-limited systems, making them highly suitable for modern computing tasks where speed and power efficiency outweigh the need for exact precision.

## REFERENCES

[1] Q. Xu, M. Todd, and S. K. Nam, “Approximate computing: A survey,” IEEE Design Test, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[2] S. Mittal, “A survey of techniques for approximate computing,” ACM Comput. Surv., vol. 48, no. 4, pp. 1–33, 2016.

[3] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Approximate computing and the quest for computing efficiency,” in Proc. 52nd Annu. Design Autom. Conf., Jun. 2015, pp. 1–6.

[4] W. Liu, F. Lombardi, and M. Schulte, “A retrospective and prospective view of approximate computing,” Proc. IEEE, vol. 108, no. 3, pp. 394–399, Mar. 2020.

[5] C. S. Lent and P. D. Tougaw, “A device architecture for computing with quantum dots,” Proc. IEEE, vol. 85, no. 4, pp. 541–557, Apr. 1997.

[6] M. Vacca et al., “Nanomagnet logic: An architectural level overview,” in Field-Coupled Nanocomputing. Cham, Switzerland: Springer, 2014, pp. 223–256.

[7] A. Khitun and K. L. Wang, “Nano scale computational architectures with spin wave bus,” Superlattices Microstruct., vol. 38, no. 3, pp. 184–200, 2005.

[8] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, “Design of approximate radix-4 booth multipliers for error-tolerant computing,” IEEE Trans. Comput., vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[9] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, “Approximate XOR/XNOR-based adders for inexact computing,” in Proc. 13th IEEE Int. Conf. Nanotechnol. (IEEE-NANO), Aug. 2013, pp. 690–693.

[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.