DESIGN AND ANALYSIS OF MAJORITY LOGIC BASED APPROXIMATE MULTIPLIER

POTUPUREDDI CHAITANYA PRIYA[1], HEMANTHNAG BOLLEPALLI[2]
[1]PG Scholar, Department of ECE, LENDI INSTITUTE OF ENGINEERING AND TECHNOLOGY, Jonnada, AP.
[2]Associate Professor, Department of ECE, LENDI INSTITUTE OF ENGINEERING AND TECHNOLOGY, Jonnada, AP.

ABSTRACT:- Approximate computing (AC) offers benefits by reducing the requirement for accuracy, thereby reducing delay. The majority logic (ML) gate functions as the fundamental logic block of many emerging nanotechnologies. These adders are designed to prevent the propagation of inexact carry-out signals to higher order computing parts to enhance accuracy. We implemented the proposed multiplier by using a unique partial product reduction (PPR) circuitry, which was based on the parallel approximate 6:3 compressor. The implemented by quantum-dot cellular automata (QCA) are analyzed to evaluate the adder designs. A significant improvement is observed over previous designs based on the experimental results. The proposed design is further designed using kogge stone adder.

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

I. 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 [12] adopts a lot of XOR operations for carry generation and propagation. However, ML operations are inefficient when representing XOR gates with two or more inputs; for more details, see Section II-A. In this article, we propose both ML-based approximate full adders (MLAFAs) and ML-based approximate multipliers (MLAMs). These contributions are described in the following.

1) Our work presents a direct method for designing multibit approximate circuits, allowing us to reduce the critical path delay and enhance the accuracy of our proposed 2- and 4-bit adders significantly. As a result of the special structure of the proposed adders, long computation sequences are less prone to accumulating errors.

2) 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.

3) Adders and multipliers are used in image processing applications. The results are evaluated by considering structural similarity (SSIM) and peak signal-to-noise ratio (PSNR). In addition, multipliers are used to develop low-power neural network (NN) accelerators for machine learning.

II. 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 [29]. 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.

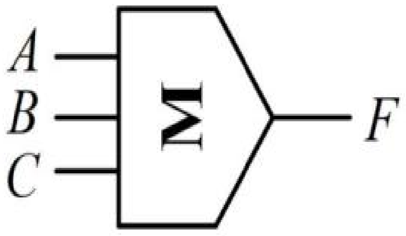


Fig. 1. Schematic of the majority gate

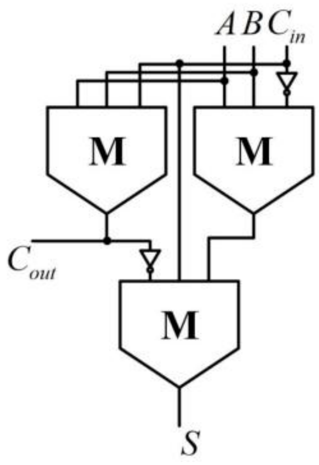


Fig. 2. Schematic of the exact full adder

III.LITERATURE SURVEY

Q. Xu, M. Todd, and S. K. Nam, “Approximate computing: A survey,”—Approximate multipliers are used in error-tolerant applications, sacrificing the accuracy of results to minimize power or delay. In this paper we investigate approximate multipliers using static segmentation. In these circuits a set of m contiguous bits (a segment of m bits) is extracted from each of the two n-bits operand, the two segments are in input to a small m×m internal multiplier whose output is suitably shifted to obtain the result. We investigate both signed and unsigned multipliers, and for the latter we propose a new segmentation approach. We also present simple and effective correction techniques that can significantly reduce the approximation error with reduced hardware costs.

We perform a detailed comparison with previously proposed approximate multipliers, considering a hardware implementation in 28 nm technology. The comparison shows that static segmented multipliers with the proposed correction technique have the desirable characteristic of being on (or close to) the Pareto-optimal frontier for both power vs normalized mean error distance and power vs mean relative error distance trade-off plots. These multipliers, therefore, are promising candidates for applications where their error performance is acceptable. This is confirmed by the results obtained for image processing and image classification applications.S. Mittal, “A survey of techniques for approximate computing,” Approximate computing trades off computation quality with effort expended, and as rising performance demands confront plateauing resource budgets, approximate computing has become not merely attractive, but even imperative. In this article, we present a survey of techniques for approximate computing (AC). We discuss strategies for finding approximable program portions and monitoring output quality, techniques for using AC in different processing units (e.g., CPU, GPU, and FPGA), processor components, memory technologies, and so forth, as well as programming frameworks for AC. We classify these techniques based on several key characteristics to emphasize their similarities and differences. The aim of this article is to provide insights to researchers into working of AC techniques and inspire more efforts in this area to make AC the mainstream computing approach in future systems.

IV. EXISTING SYSTEM

The majority gate performs a multi-input logic operation and is shown in Fig. 1; the inputs are A, B and C and the output is F.

The logic expression of the 3- input majority gate (voter) is given by

$$F = M(A,B,C) = AB + BC + AC$$

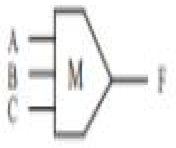


Fig.3. Majority gate (3-input voter)

Fig. 3 Majority gate (3-input voter) Research on the design of ML approximate circuits has only recently been pursued; In the authors proposed a one-bit full adder circuit. In this paper, we propose a few ML based approximate full adders; one-bit as well as multi-bit approximate adders are considered. one-bit approximate full adder (AFA1), AFA1produces the output S as the complement of C but introduces 2 errors when computing the output S

$$C_{out} = M(A,B,C)$$
$$S = \overline{C_{out}}$$

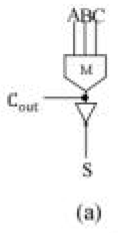


Fig 4. One-bit approximate full adder AFAI

One-bit approximate full adder is presented and compared with one-bit accurate full adder. A new one-bit approximate full adder, namely, AFA2. One-bit approximate full adder is presented and compared with one-bit accurate full adder. A new one-bit approximate full adder, namely, AFA2.

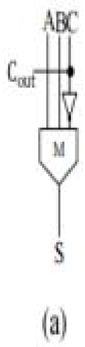


Fig 5. Schematic one-bit approximate full-adder AFA2

They show better overall performance than the other two designs. The eight-bit approximate adders are shown in Fig.5; The designs significantly reduce the number of gates and delay but at the cost of a decrease in accuracy.

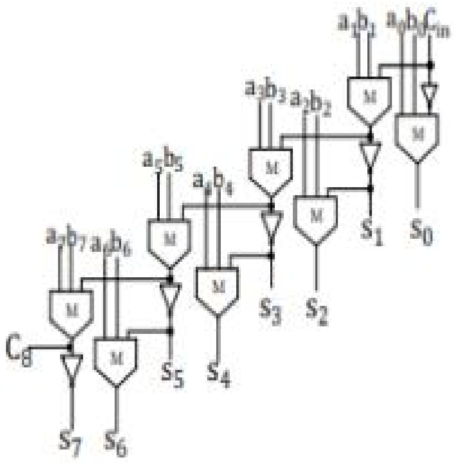


Fig 6. Existing system output

V.PROPOSED APPROXIMATE MULTIPLIER

A. Design of Half-Adder

The half adder is a modest combinational circuit that executes the addition of two bits. The half adder circuit is traditionally designed using EXOR and AND gates. The addition of two numbers A and B processed and the respective outputs are Sum and Carry. From the concept of truth table of the half adder as in Table 1, one can recognize that the Sum output is 1 when either of the inputs (A or B) is 1, and the Carry output is 1 when both the inputs (A and B) are 1. Figure 1 shows the general diagram and Table 1 truth table of the half adder circuit. The logic function of the half adder is,

Sum = A’B + AB’

Carry = AB

The QCA representation for the above equation based on MG is,

Sum = M (M (A,B’,0), M(A’,B,0), 1)

Carry = M (A,B,0)

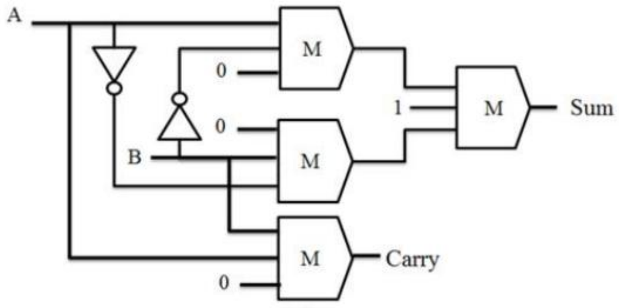


Fig7: half adder using majority gates

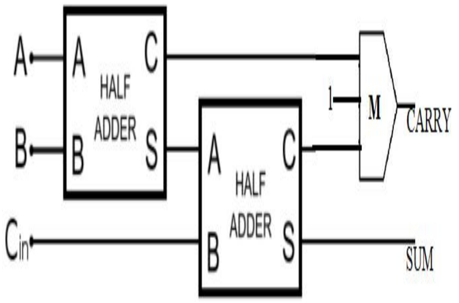


Fig 8: full adder using half adder

B. Proposed Approximate Multipliers

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.

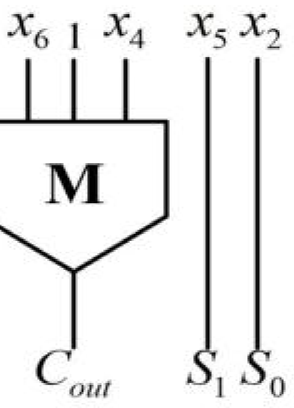


Fig. 9. Schematic of the approximate parallel 6:3 compressor

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.

C. 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 RCA 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. 9 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.

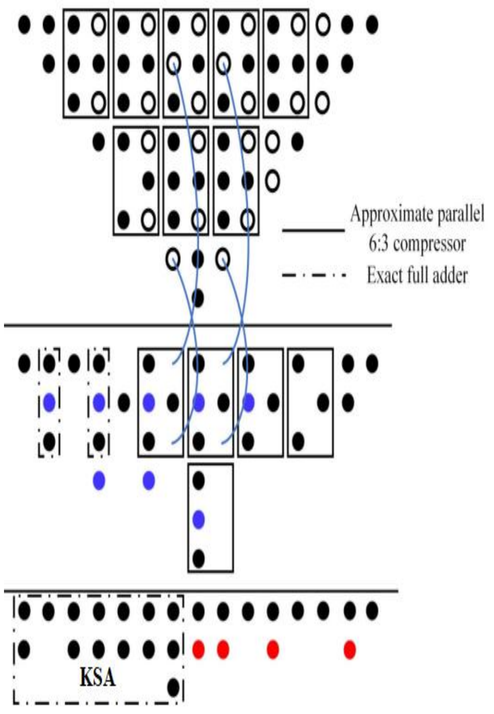


Fig 10 : Reduction process of an unsigned 8 × 8 multiplier proposed PPR circuit1

D. KOGGE Stone Adder

KSA is a parallel prefix form carry look ahead adder. It 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 KSA, carries are computed fast by computing them in parallel at the costof increased area.

The complete functioning of KSA can be easily comprehended by analyzing It in terms of three distinct parts :

1. Pre processing

This step involves computation of generate and propagate signals corresponding too each pair of bits in A and B.

pi = Ai *xor* Bi

gi = Ai *and* Bi

2. Carry look ahead network

This block differentiates KSA from other adders and is the main force behind its high performance. This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals .

Pi:j = Pi:k+1 *and Pk:j*

Gi:j = Gi:k+1 *or* (Pi:k+1 *andGk:j*)

3. Post processing

This is the final step and is common to all adders of this family (carry look ahead). It involves computation of sum bits.

Si = pi xor Ci-1

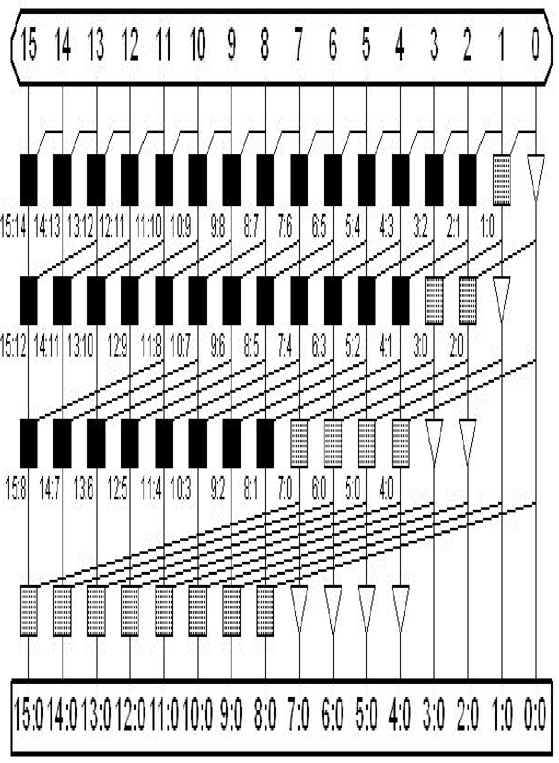


Fig11: 16 bit kogge stone adder

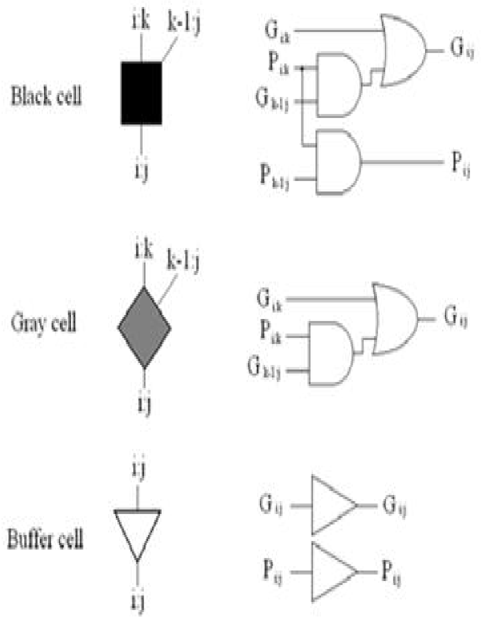


Fig 12: Complex logic cells inside the Prefix Carry Tree

VI. 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.

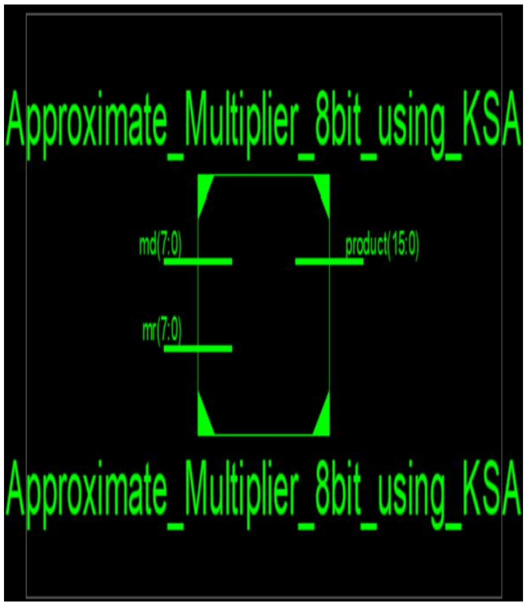

Approximate_Multiplier_8bit_using_KSA
md(7:0)
product(15:0)
mr(7:0)
Approximate_Multiplier_8bit_using_KSA

Fig13: RTL Schematic of proposed design

TECHNOLOGY SCHEMATIC:- The technology schmatic makes the reesentation 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.

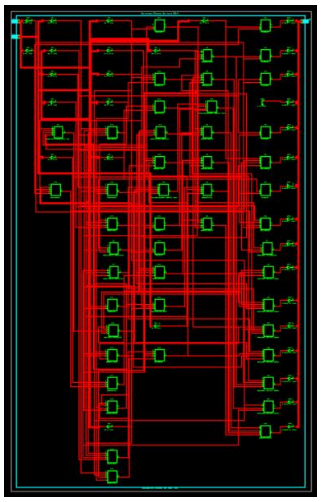
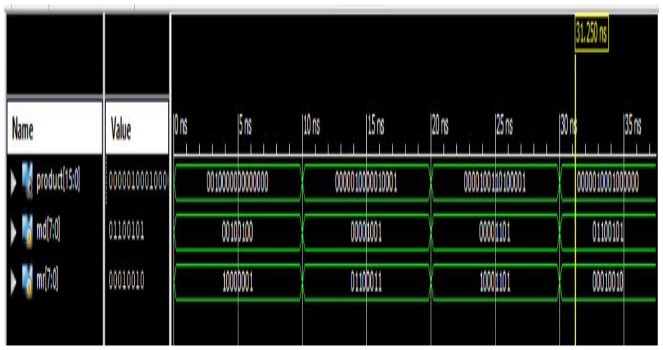


Fig14: View Technology Schematic of proposed design

SIMULATION:-The simulation is the process which is termed as the final verification in respect to its working where as 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.



```
product[15:0]  0000010001000   0010000000000000  0000010000010001  0000100110100001  0000010001000000
md[7:0]        01100101        00100100          00001001          00001101          01100101
mr[7:0]        00010010        10000001          01100011          10001101          00010010
```

Fig15: 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	Existed approximate multiplier	proposed approximate multiplier
Delay(ns)	**14.078**	**12.568**

Table1 : delay comparison

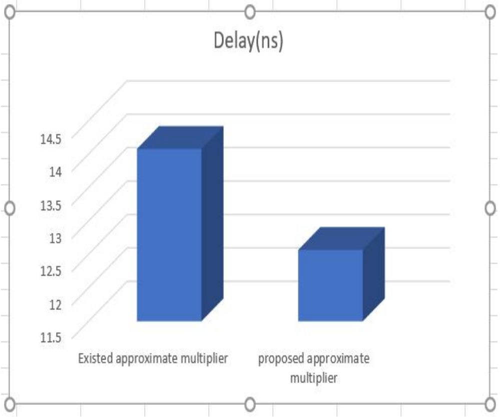


Fig 16: Delay comparison bar graph

VII. CONCLUSION

In this project presents the designs, analysis a novel approximate 6:3 compressor and a unique PPR circuit are proposed for the parallel compressor in multiplier using kogge stone adder. They are able to reduce the delay without significantly degrading the quality of the multiplier. In addition, the proposed compressor is strongly generalizable. It requires only one majority gate with a constant of "1," which is an OR gate that is excellently implemented in other logic primitives. Compared with other existing approximate designs, there is a significant improvement in terms of delay.

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. EE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.