



# ANALYZING AND DESIGNING MAJORITY-BASED APPROXIMATE ADDERS AND MULTIPLIERS

<sup>1</sup>Vuyyala Shankar,<sup>2</sup>Yellamelli Jalajakshi,<sup>3</sup>Thatiparthi Mounika,<sup>4</sup>Mudragadda Sheshi Rekha

<sup>1,2,3,4</sup>Assistant Professor

Department of ECE

Samskruti College of Engineering and Technology, Hyderabad

## ABSTRACT:

Approximate computing is a new paradigm for nanoscale technologies that overcomes the problem of mistake tolerance in computation, allowing for greater performance with less power. The 3-input MV is an important building component of majority logic (ML) in digital circuit design and is expected to play a significant role in numerous emerging nanotechnologies. In order to do this, the suggested multipliers and adders employ approximation compressors and a reduction circuitry that makes use of so-called complement bits. An approach is provided for selecting appropriate complement bits, and a size-dependent multiplier-dependent effect factor is developed and studied. The proposed designs are evaluated based on their feasibility using hardware metrics (such delay and gate complexity) and error metrics. The proposed designs are proved to be superior than existing ML-based systems in the literature. Case studies of bug-free programs are offered to show that the suggested architectures work as advertised.

**Keywords:** Majority logic, approximate adder, approximate multiplier, complementbits, approximate compressor, image processing.

**DOI Number:** 10.48047/nq.2022.20.8.nq221141

**NeuroQuantology 2022; 20(8): 11079-11087**

## 1. INTRODUCTION

Even though alternative nanoscale technologies have been proposed to replace CMOS after the expiration of Moore's law, the problem of power dissipation has not yet been overcome, despite the fact that the integration density of nano electronics devices is continuing to increase at a rapid pace. The use of approximate computing has the potential to reduce the negative effects that computers has on the environment while simultaneously improving system performance in error-tolerant applications such as multimedia signal processing, machine learning, and pattern recognition. Complementary metal-oxide semiconductor (CMOS) based approximation computer arithmetic circuits have been the subject of a significant number of research investigations. It has been shown that it is possible to create approximation adders, multipliers, and dividers for both fixed-point and floating-point formats. Error metrics such as mean error distance (MED), normalized mean error distance (NMED), and relative mean error distance (RMED) are only a few examples of the types of error

metrics that have been established to investigate the errors that are caused by approximate arithmetic circuits.

Due to the fundamentally distinct logic structure of certain emerging technologies, such as quantum computing assistants (QCA), nanomagnetic logic, and spin wave devices, it will be impossible for these technologies to quickly adopt the approximation designs used in CMOS circuits. The majority logic (ml) method of operation is the standard method of operation that is used in the modern electronics industry. In contrast to this, one could use the more standard Boolean logic. The functionality of a majority gate, which is a multi-input logic device, is illustrated in figure 1;

$$= M(A, B, C) = AB + BC + AC \quad (1)$$

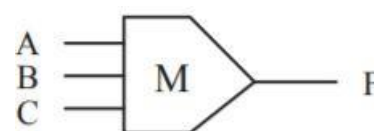


Fig.1.Majoritygate(3-input voter).

Just recently, research and development efforts have



been focused on small approximation circuits. For instance, a full adder circuit that requires only a single bit of logic has been produced in recent years. Adders with one bit and several bits of approximation are discussed, and certain ml-based approximation full adders are presented. Case studies conducted with Qca are used to demonstrate that the techniques that were proposed are practical. Both fault-fixing and hardware analysis are included in this course.

**2. RELATED WORK**

**Quantum-dot Cellular Automata**

In contrast to CMOS, the polarization state of cells is used to encode binary information in QCA, and the contacts between cells are managed by the Coulombic forces in order to implement the functionality of the circuit. In Figure 2a, we can see the components that make up a QCA cell, which are as follows: four quantum dots and two tunneling electrons. Due to the fact that charges of the same type repel one another, electrons can only exist at the corners (also known as "points") of diagonals. This results in two polarization states for each cell, with -1 and +1 corresponding to the logical values 0 and 1, respectively. These two states are referred to as positive and negative, respectively. The QCA protocol requires a clocking system that has four separate phases: switch, hold, release, and relax. These phases are used to govern the flow of data. When a majority gate is used in QCA, as can be seen in Figure 2b, a 0.25-clock delay is introduced into each clocking zone. This is the case anytime the majority gate is used. Another common type of basic gate that may be seen in QCA is an inverter, which can be shown in Fig. 2c. Additional information, with a focus on the clocking system and the QCA technology, is presented here.

**MLbasedOne-BitExactandApproximate Full Adders**

Figure 3 depicts an ML-based full adder that needs only three majority gates and two inverters to provide one-bit precision. The inputs for the one-bit adder are the letters A, B, and C, and it outputs the characters S and C respectively. When it comes to the products that C and S create, they look like this:

$$C_{out} = AB + BC + AC = M(A, B, C) \tag{2}$$

$$S = A \oplus B \oplus C = M(\overline{C_{out}}, M(A, B, \overline{C}), C) \tag{3}$$

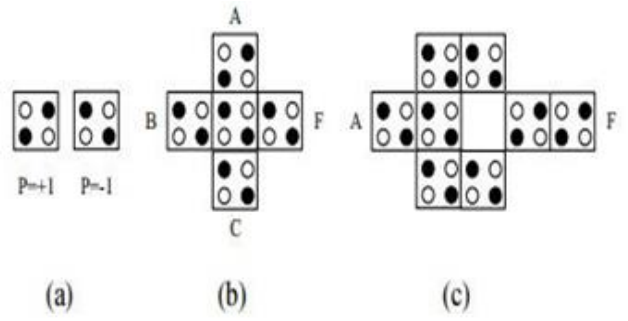


Fig.2. QCA basic elements: (a) QCA cell, (b) QCA majority gate (voter), and (c) QCA inverter

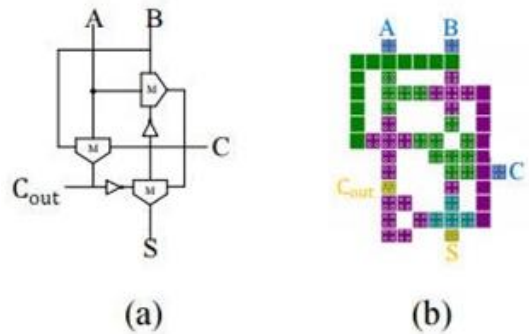


Fig.3. One-bit accurate full adder: (a) schematic of accurate adder, (b) layout of accurate adder in QCA  
 Figure 4 from Labrado et al. presents a picture of their proposed design for the one-bit approximation full adder (AFA1). The output S, which is the complement of C, is calculated improperly by AFA1 for four of the eight possible input combinations (see Table I for further information).



Fig. 4. One-bit approximate full adder: (a) schematic of AFA1, (b) layout of AFA1 in QCA.

In the truth table, we highlight the circumstances in which the outcomes of the approximation full adder are distinct from the outcomes of the exact full adder. The equations for carry and sum that correspond to this are as follows.

$$C_{out} = M(A, B, C) \tag{4}$$

$$S = \overline{C_{out}} \tag{5}$$

**Error Metrics for Approximate Circuits**

As approximate computing introduces errors, error metrics are required to evaluate the accuracy of approximate circuits. In this paper, we evaluate approximate designs using the MED and the NMED. The MED is defined as



the average of the Error Distance (ED) which is the absolute difference between the approximate and the accurate results across all possible inputs. The NMED is the normalized MED. The definitions of ED, MED and NMED are as follows: where X, Y, and MAX denote the accurate result, the approximate result, the counts of all possible inputs and the maximum value of the result, respectively.

[1] SYSTEM DESIGN

In this paper, we show the state-of-the-art one-bit exact full adder and the state-of-the-art one-bit approximation full adder and discuss the advantages and disadvantages of each.

**Proposed One-bit Approximate Full Adder**

In this research, we introduce AFA2, a new full adder that approximates with a single bit. The truth table in Table I shows that C is quite similar to C in all but two of the eight possible input configurations. To compute the carry out of a one-bit full adder (as opposed to an accurate one-bit full adder), C is roughly comparable to the number of gates you would need to conserve.

**Comparison and Discussion**

The ratio of majority gates to inverters, as well as other metrics like delay, area, and minimum delay, are taken into account. Table II shows a comparison between the exact adder AFA1 and the proposed approximation adder AFA2. AFA2 can save design size by up to 72% compared to the accurate adder, while also saving 2 majority gates, 1 inverter, and 0.25 clock cycles. In addition, AFA2 has a smaller impact on QCA than AFA1.

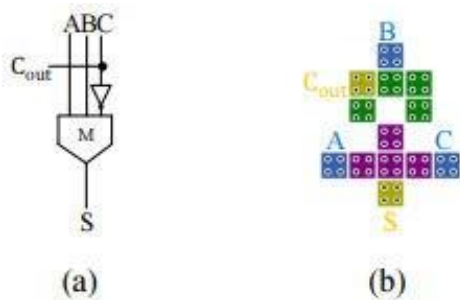


Fig.5. Proposed one-bit approximate full adder: (a) schematic of AFA2, (b) layout of AFA2 in QCA.

Table 1: Truth Table of 1-bit MLAFAs

Inputs			Accurate		AFA1[3]		AFA2	
A	B	C	C <sub>out</sub>	S	C <sub>out</sub>	S	C <sub>out</sub>	S
0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	1	0
0	1	0	0	1	0	1	0	1
0	1	1	1	0	1	0	1	0
1	0	0	0	1	0	1	0	1
1	0	1	1	0	1	0	1	0
1	1	0	1	0	1	0	0	1
1	1	1	1	1	1	0	1	1

Table 2: Comparison of 1-bit MLAFAs

ADDER TYPE	MV	INV	D	A(nm <sup>2</sup> )	MED	NMED
Accurate [14]	3	2	0.5	60050	0	0
AFA1[3]	1	1	0.25	18902	0.25	0.083
AFA2	1	1	0.25	16284	0.25	0.083

**Proposed multi-bit approximate adders**

The proposed and existing one-bit approximate full adders are proposed to be combined to create multi-bit approximation full adders. Hardware architectures and error metrics are compared and analyzed.

**Proposed two-bit approximate adders:**

When the following values are input into a two-bit adder, the output is "care": a = aa, b = bb, c = cc, and s = ss. The two-bit approximation full adder can be built in one of four ways, all of which involve connecting one-bit approximate full adders (AFA1 and AFA2) in series. Cascaded AFA1 yields the 2-bit AFA11 structure. The AFA22 architecture is, in essence, a string of AFA2s. Both AFA12 and AFA21 use AFA2 to get the LSB, although only one of the two components, AFA1, is used to do so. Figure 7 depicts the schematics for these two-bit approximation QCA adders.

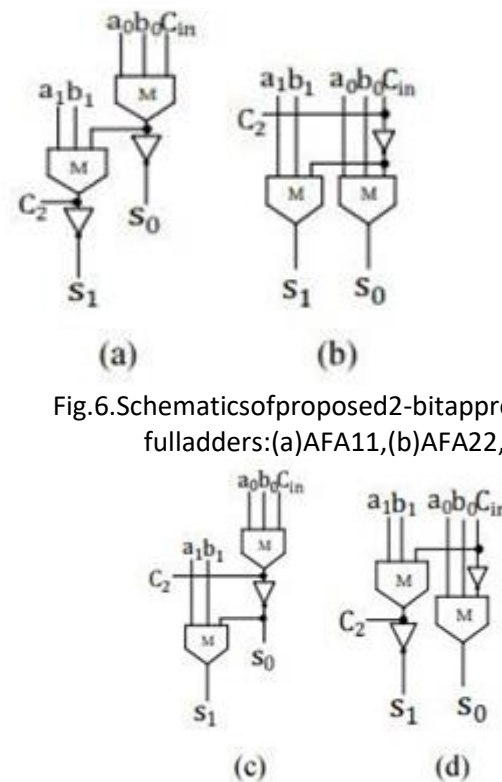


Fig.6. Schematics of proposed 2-bit approximate full adders: (a) AFA11, (b) AFA22,

(c) AFA12 and (d) AFA21.



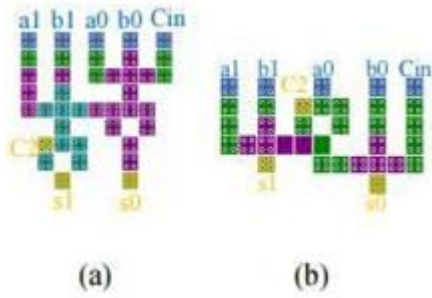
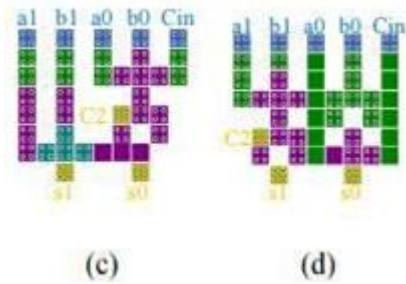


Fig.7. Layouts of proposed 2-bit approximate full adders in qca: (a) AFA11, (b) AFA22, (c) AFA12 and (d) AFA21.



(c) AFA12 and (d) AFA21.

Table III demonstrates that the suggested two-bit approximation adders produce incorrect results for 14 of the 32 possible input combinations. Cascading two identical one-bit approximation full adders yields a larger med and NMED than does cascading two different ones, but AFA22 occupies less physical space and has lower latency than AFA12. Because it uses fewer gates, AFA12 can be implemented with just one fewer inverter than AFA21 does. Since AFA21 has lower latency than AFA12, it requires only 0.25 clocking zones less.

Table3: Comparison of 2-bit MLAFAs

ADDER TYPE	MV	INV	D	A(nm <sup>2</sup> )	MED	NMED
AFA11	2	2	0.5	44073	0.75	0.107
AFA22	2	1	0.25	35383	0.75	0.107
AFA12	2	1	0.5	39267	0.625	0.089
AFA21	2	2	0.25	41069	0.625	0.089

Both the area-delay product and the NMED are considered in the comparative findings shown in Fig. 8, with AFA21 being the best design because of its proximity to the origin. When compared to AFAs and AFAs with a single type of approximate full adder, AFA12 and AFA21 (with multiple varieties of one-bit approximation full adders) perform better.

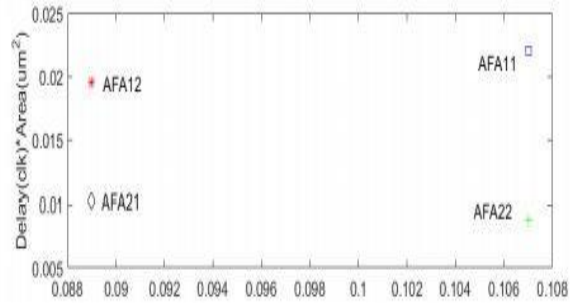


Fig. 8. Evaluation of proposed 2-bit approximate full adders (NMED Vs delay area product)

**Proposed four-bit approximate adders**

A 4-bit approximation full adder can be built by cascading two 2-bit approximation full adders. The AFA12 and AFA21 two-bit approximation full adders are chosen because of how well they function. Figure 9 depicts the four potential permutations, and Figure 10 depicts the QCA representation of these permutations. Despite sacrificing precision, Table IV shows that the proposed designs lower the number of gates required to implement an adder. There is potential for a 67% reduction in size and a 50% reduction in latency. The AFA1221 has the greatest med/NMED while having the fewest gates and the quickest latency. The med/NMED ratio of AFA2121 and AFA2112 are identical, however the delay in AFA2121 is much smaller. When compared to its predecessor, the AFA2112, the AFA1212 uses less power and thus fewer inverters.

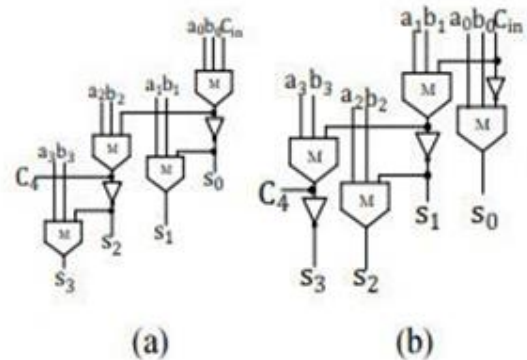
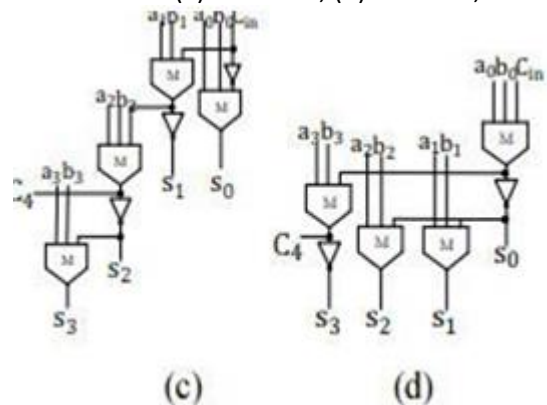


Fig.9. Schematics of proposed 4-bit approximate full adders: (a) AFA1212, (b) AFA2121, (c) AFA1221, (d) AFA2112.



(c)AFA2112and(d)AFA1221.

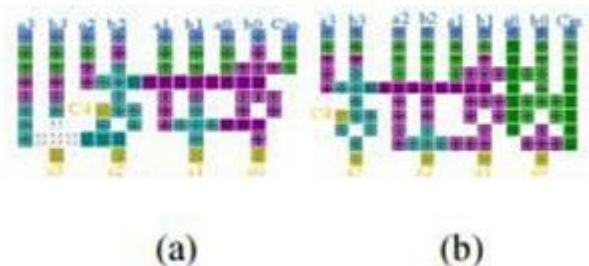
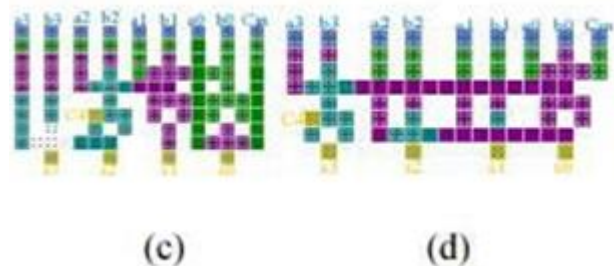


Fig.10. Layout of proposed 4-bit approximate full adders in qca: (a)AFA1212, (b) AFA2121,



(c) AFA2112 and (d) AFA1221.

Figure 11 shows how dominant AFA2121 is. AFA2121 has a sizable NMED, while AFA1212, AFA2121, and AFA2112 all have sizable NMEDs but must be delayed for a shorter amount of time. Two of the same kind of the suggested two-bit approximation full adders cascaded together is the best approach for four-bit systems since it provides higher performance than cascading multiple kinds of approximate full adders.

Table 4: Comparison of 4-bit MLAFAs

ADDER TYPE	MV	INV	D	A(nm <sup>2</sup> )	MED	NMED
CFA4[14]	12	8	1.5	405000	0	0
RCA4[15]	12	4	1.75	254200	0	0
AFA1212	4	2	0.75	76120	2.83	0.091
AFA2121	4	3	0.5	82619	2.87	0.092
AFA2112	4	3	0.75	83936	2.87	0.092
AFA1221	4	2	0.5	82085	5.45	0.175

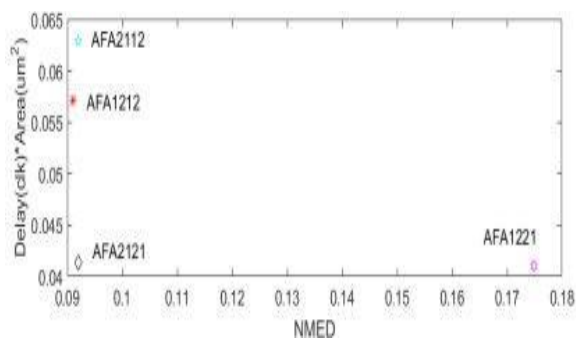
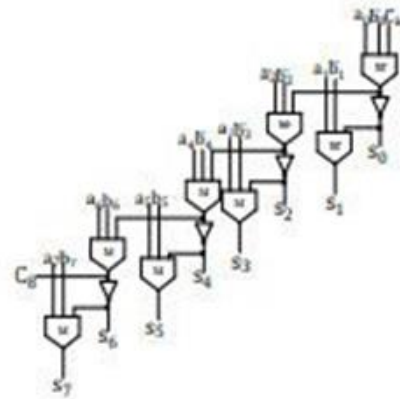


Fig.11. Evaluation of proposed 4-bit approximate full adders (NMED Vs delay area product)

**Proposed eight-bit approximate adders**

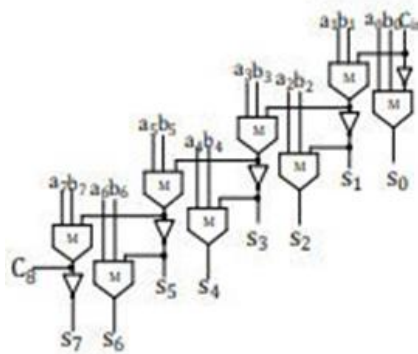
Our improved eight-bit approximate adders are a result of cascading two four-bit approximate adders,

which achieves the same greater overall performance as the AFA1212 and AFA2121 designs. Figure 12 depicts the proposed eight-bit approximation adders, while Figure 13 depicts their Qca implementations. Table V displays all comparison information. Designs are offered that sacrifice precision in order to significantly reduce both the number of gates and the latency. In comparison to the others, the AFA2121-2121 and AFA1212-2121 adders use fewer gates and have a smaller delay, whereas the AFA1212-1212 and AFA1212-2121 adders just need one fewer inverter.



(a)

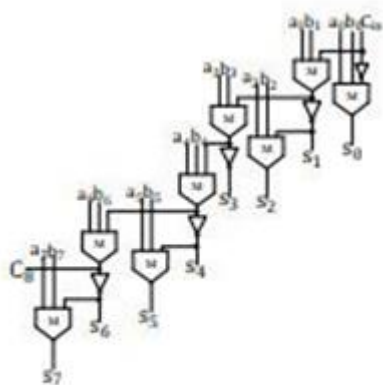
Fig. 12. Schematics of proposed 8-bit approximate full adders: (a) AFA1212-1212,



(b)

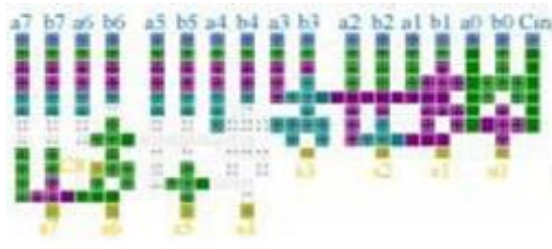
(b) AFA2121-2121,





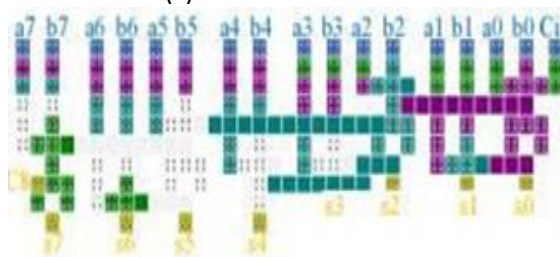
(c)

(c)AFA2121-1212and



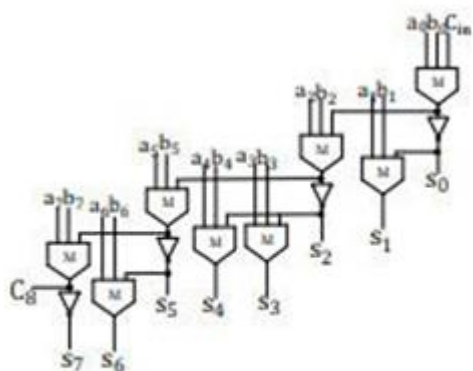
(c)

(c)AFA2121-1212and



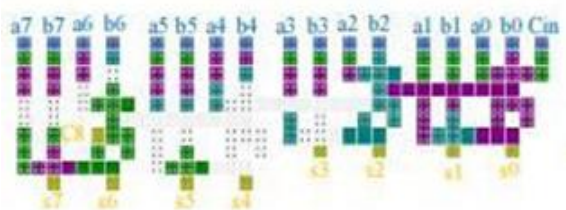
(d)

(d) AFA1212-2121.



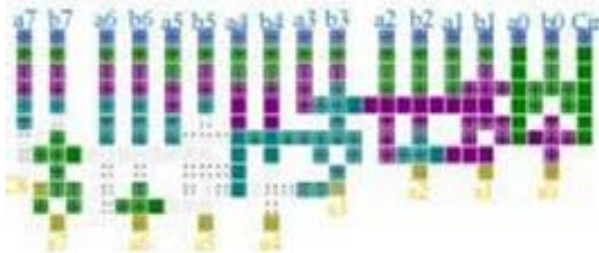
(d)

(d)AFA1212-2121.



(a)

Fig.13.Layoutsofproposed8-bitapproximatefulladdersinQCA:(a)AFA1212-1212,



(b)

(b)AFA2121-2121,

Table5:Comparison of 8-bitMLAFAs

ADDER TYPE	MV	INV	D	A(nm <sup>2</sup> )	MED	NMED
CFA8[14]	24	16	2.5	948700	0	0
RCA8[15]	24	8	2.75	745200	0	0
AFA1212-1212	8	4	1.25	184640	46.20	0.090
AFA2121-2121	8	5	1	206425	47.02	0.092
AFA2121-1212	8	5	1.25	218257	46.40	0.091
AFA1212-2121	8	4	1	191126	46.82	0.092

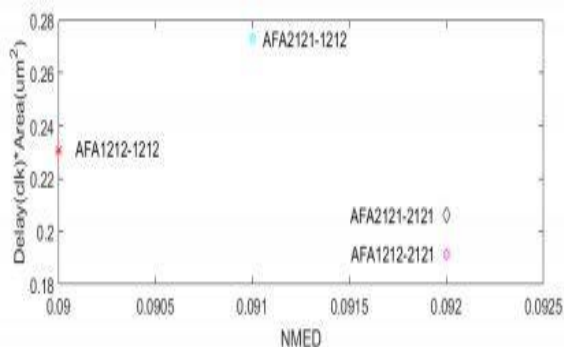


Fig. 14. Evaluation of proposed 8-bitapproximate full adders (NMED vs delay are a product)

**ML based approximate multipliers**

In this section, we examine 22 MLAMs and how they can be used to compute approximation multipliers based on the ml notation. In order to compensate for mistakes, a selection method may use a so-called complement bit. As shown in Figure 7, the following steps should be followed while designing nn MLAMs. The multiplicand  $a_n1234 \dots a_210$  and the multiplier  $b_n1234 \dots b_210$  are each divided into  $n/2$  modules (each of 2 bits as a unit), which are then substituted



into the equation, and the compensating bits are added as determined by the size of the multiplier. Then, the partial product reduction (PPR) hardware is used, which is a compression method that can be either precise or approximative. The probability of obtaining the pp of two rows (or a carry in the lowest order) depends on the distribution of the PPSs generated and the compensation bits. Depends on how good an adder the last person is.

**2x2-MLAM**

As can be seen in the ML description of the 2 2 am design, output 1 requires three majority gates, one more than output 0 and two more than output 2.

$$out_0 = M(A_0, B_0, 0)$$

$$out_1 = M(M(A_1, B_0, 0), M(A_0, B_1, 0), 1)$$

$$out_2 = M(A_1, B_1, 0)$$

Problems that swiftly proliferate as the design's scope grows cannot be tolerated, thus there is room for improvement. One is used as the out1 of the 2 2 MLAM because of these considerations, while the other is used as the compensatory bit. In this study, we focus on one specific incident. Similar procedures are followed for the other situation.

$$out_1 = M(A_0, B_1, 0)$$

$$\Delta = M(A_1, B_0, 0)$$

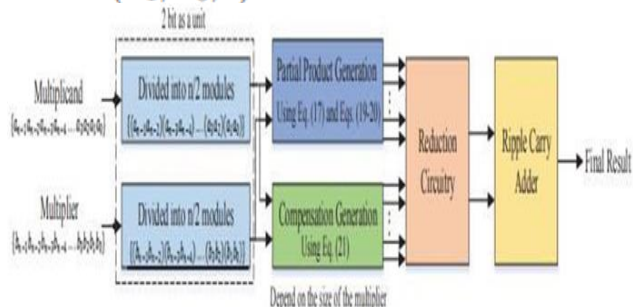


Fig.15. Proposed design flow of n x n MLAMs.

By splitting the operands into numerous units, with 4 signifying a complement bit, the 2 2 MLAM can be used as a building block to generate larger multipliers. The 44 MLAMs with all complement bits that require further reduction are depicted in Figure 15, and the 88 MLAMs with all complement bits are depicted in Figure 16.

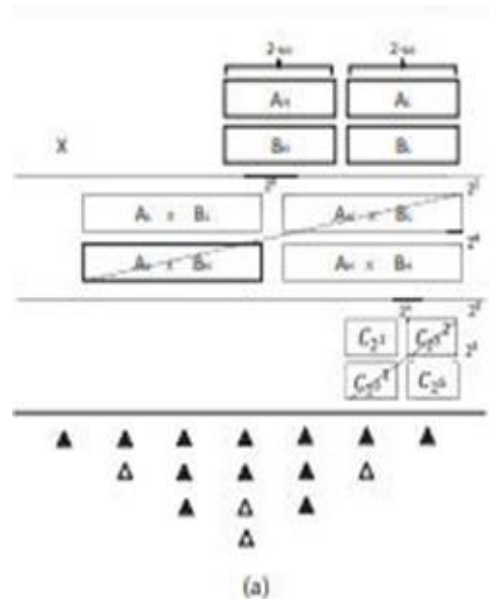
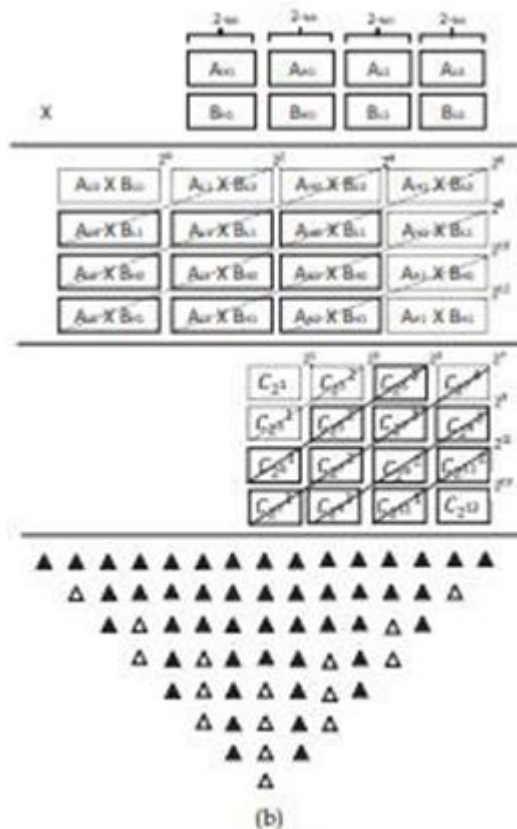


Fig. 16. PPG and complement bitgeneration of MLAMs: (a) 4x4 MLAM, and



(b) 8 x 8 MLAM.

**3. RESULTS**

The suggested QCA approximate full adders are created and evaluated on the one-bit example using the XILIN ISE software. The XILIN ISE program was developed at the University of Calgary in order to construct and simulate QCAs. The simulation and design process will go something like this. To get started, a design for the suggested one-bit



approximate complete adder is generated. We build multi-bit adders by expanding on the original one-bit concept.

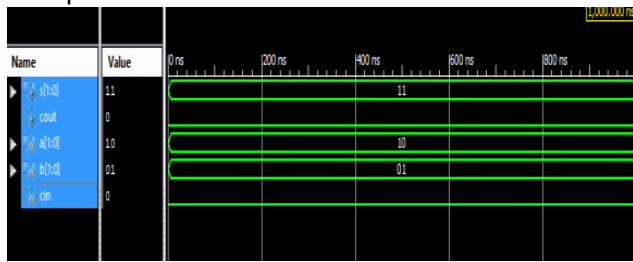


Fig17AFA11

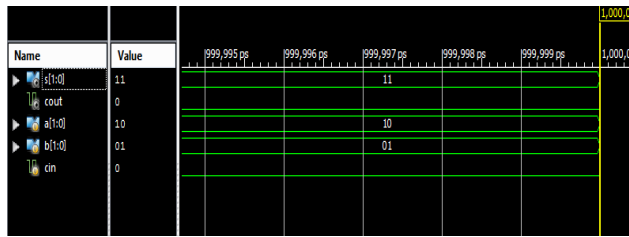


Fig18AFA22

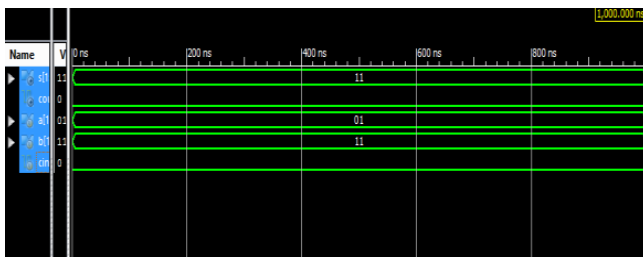


Fig19AFA21

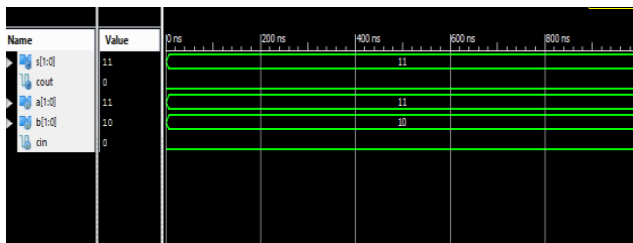


Fig 20AFA12

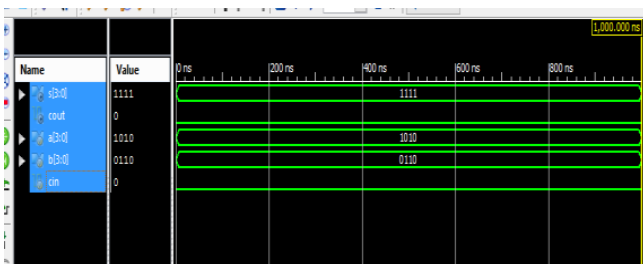


Fig21AFA1212

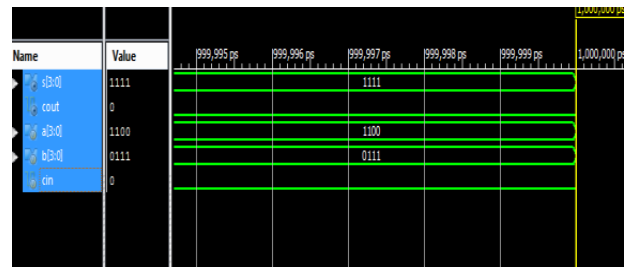


Fig22 AFA 1221

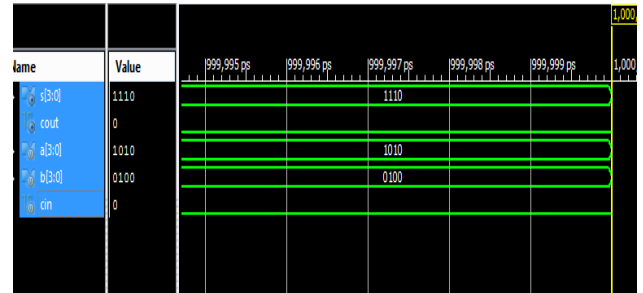


Fig23 AFA 2112

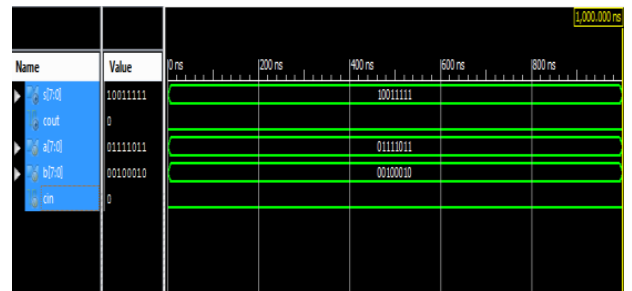


Fig24 AFA12122121

11086

4.

5. CONCLUSION

Approximate adders and multipliers based on majority logic are presented, implemented, and evaluated in this article. Multiple-bit, two-bit, and one-bit AFAs that use ML have been proposed because they are simpler to implement and take up less processing time without sacrificing too much accuracy. Different MLACs, such as those using MLAFAs or K-Map simplification, and approximate PPR circuitry have been proposed as part of ML-based multi-bit AMs; an influence factor has been defined to measure the importance of various complement bits; a thorough analysis based on the size of multipliers has been pursued in selecting the complement bits; etc.

REFERENCES

[2] S. L. Lu, "Speeding up processing with approximation circuits," Computer, vol.37, no. 3, pp. 67-73, 2004.  
 [3] J.Han and M.Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in Proc. European Test Symposium, pp.



- 1-6,2013
- [4] S. Hashemi, R. Bahar, and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," in Proc. IEEE/ACM International Conference on Computer Design, pp. 418-425, 2015.
- [5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans Computer-Aided Design of Integrated Circuits and Systems, vol. 32, pp. 124-137, 2013.
- [6] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-space exploration of approximate multipliers," in Proc. Int. Conf. Computer-Aided Design, pp. 1-6, 2016.
- [7] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Transactions on Computers, vol. 64, no. 4, pp. 984 - 994, 2014.
- [8] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans Computers, vol. 62, no. 9, pp. 1760-1771, 2013.
- [9] K. Walus and G. Jullien, "Design tools for an emerging SoC technology: quantum-dot cellular automata," in Proc. IEEE, vol. 94, no. 6, pp. 1225-1244, 2006.
- [10] C. Lent and P. Tougaw, "A device architecture for computing with quantum dots," in Proc. IEEE, vol. 85, no. 4, pp. 541-557, 1997.
- [11] M. Vacca, M. Graziano, J. Wang, F. Cairo, G. Causapruno, G. Urgese, A. Biroli, and M. Zamboni, "Nanomagnet logic: an architectural level overview," Lecture Notes in Computer Science, pp. 223-256, 2014.
- [12] A. Khitun and K. L. Wang, "Nanoscale computational architectures with spin wave bus," Superlattices and Microstructures, vol. 38, no. 3, pp. 184-200, 2005.
- [13] C. Labrado, H. Thapliyal, and F. Lombardi, "Design of majority logic based approximate arithmetic circuits," in Proc. IEEE Int. Symp. Circuits and Systems, pp. 2122-2125, 2017.
- [14] Srinivas, L., & Umapathi, N. (2022, May). New realization of low area and high-performance Wallace tree multipliers using booth recoding unit. In AIP Conference Proceedings (Vol. 2393, No. 1, p. 020221). AIP Publishing LLC.
- [15] N. Umapathi, G. M. Krishna and L. Srinivas, "A Comprehensive Survey on Distinctive Implementations of Carry Select Adder," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1-5,
- [16] Murali Krishna G., Karthick G., Umapathi N. (2021) Design of Dynamic Comparator for Low-Power and High-Speed Applications. In: Kumar A., Mozar S. (eds) ICCCE 2020. Lecture Notes in Electrical Engineering, vol 698. Springer, Singapore.
- [17] N. Umapathi, G. L. 2020. Design and Implementation of Low Power 16x16 Multiplier using Dadda Algorithm and Optimized Full Adder. International Journal of Advanced Science and Technology. 29, 3 (Feb. 2020), 918 - 926.
- [18] Prasad, R., Umapathi, N., & Karthick, G. (2022). Error-Tolerant Computing Using Booth Squarer Design and Analysis. Specialusis Ugdymas, 2(43), 2970-2985.
- [19] Saikrishna, D., Umapathi, N., & Mothe, S. (2022). Delays in the Generation of Test Patterns and in the Selection of Critical Paths. Specialusis Ugdymas, 2(43), 2986-2997.
- [20] Swarnalatha, B., & Umapathi, N. (2022). Voltage over Scaling-Based Dadda Multipliers for Energy-Efficient Accuracy Design Exploration. Specialusis Ugdymas, 2(43), 2942-2956.
- [21] Pranitha, G., Karthick, G., & Umapathi, N. (2022). Using a Configurable Floating Point Multiplier to Trade-Off Runtime Efficiency and Accuracy. Specialusis Ugdymas, 2(43), 2957-2969.
- [22] Guguloth Arjun, Umapathi, N. (2022). Design of Low-Area and High-Performance Radix - 4 Booth Multipliers. High Technology Letters, Volume 28, Issue 10, 100-106.

