A Critical Analysis of Approximate Adders: Correctness and Analysis of Performance

Ajay Kumar Gottem, Arunmetha S, Aravindhan Alagarsamy, Murali Krishna B
Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, AP 522 501, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract—In digital Very Large-Scale Integration (VLSI) the adder is a basic computational block for many circuits. Approximate adders were proposed as feasible solution in error-tolerant applications to provide a proper trade-off with accuracy to have better performance parameters in terms of energy, area, and delay. The state of the art of approximate adders are demonstrated in order to greatly enhance the operational features. To obtain the greatest number of approximation advantages, in this paper a systematic comparison between different approximate adders is presented. Highlighted Approximate Adder as a root map for various applications, which is more useful for some of the researcher's work in this field. This paper looks at existing estimated adder designs and compares them in terms including both error and circuit characteristics. Four different Approximate adders are reviewed in terms of Area, Delay, Simulation time and device utilization summary. Gate level implementation of selected adders are described in detail. The cost functions of selected Approximate adders are compared against various FPGA standard architectures. This paper looks at existing estimated adder designs and compares them in terms including both error and performance analysis. Comparison Results indicate an average of 49% change in Area Delay Product (ADP) and 6% variation in Simulation time.

Index Terms— Approximate adder, Area delay product (ADP), Field programmable gate array (FPGA), Parallel prefix adder, Very Large-Scale Integration (VLSI).

INTRODUCTION

In the design of digital circuits, the key goals are to reduce power consumption, optimize area and increase speed. Approximate adder is used as a basic element in almost all arithmetic operations such as multiplication, subtraction, and division in digital circuits. Designers have been paying a lot of attention to approximate adders in recent years since they can be used in a variety of applications where some error is appropriate [1]-[5]. Because of the high computation demand in applications, approximate adders have grown in popularity [6] such as machine learning, deep learning, image processing and digital signal processing (DSP), wireless communication and search engines. We intentionally introduce errors in these applications to take advantage of delay, area, and power.

Therefore, approximate adders with significant loss have been implemented because we do not require a single golden result for those applications, but rather a sufficient result. Approximate computing was implemented at various abstraction levels like dynamic-voltage-accuracy-frequency-scaling in [7], computation at algorithmic level in [8] and circuits are redesigned into approximate variants by inserting prediction logic designed specifically for arithmetic adders.

The approximate adders are, one with approximate adder cell and other with Han-Carlson adder is implemented. The proposed approximate adder is divided into non-overlapped summation blocks in which carry from the previous block is independent to the next summation block. The carry input from each block is dependent on the input operands itself, but not from the previous blocks. Carry chain propagation is truncated to adder block itself, in worst case carry propagated to next corresponding block, that reduces delay drastically. Approximate Adder cells (AAC) and Error Recovery Unit (ERU) which are widely used in Approximate Computing (AC) that are clearly described in the further sections of this paper.

This paper is structured as follows. Section II explains about Literature Survey. Section III highlights the Comparative Evaluation of State-of-Art Approximate adders. Finally, Discussions and Conclusion in section V.

LITERATURE SURVEY

In this section, we will briefly address current works that use a state-of-the-art approximate adder. The work in [9] is a reconfigurable approximate carry look ahead (CLA) adder that can be used as both an approximate and an exact adder. It implements the specification with an exact carry look ahead adder, as well as multiplexers to pick the accuracy configurability and function as an exact and approximate adder. In this control input is used to vary the function which will act as either approximate adder or exact adder. This method of implementation would result in increased hardware area complexity and delay.

The work in [10] implemented an estimated carry skip adder that is implemented using an exact carry skip adder. To reduce the overall critical path delay, the n-bit adder is divided into l-bit sub adders, with each sub adder receiving the carry from the carry prediction block. In order to increase accuracy and minimize error rate, an

additional error magnitude correction block is used. As a result, adding this additional block to the estimated adder design would increase the critical path delay and power consumption. The design's cost function is affected.

A high-accuracy block-based carry speculative adder (BCSA) is proposed in [11], which is implemented using a carry propagate adder and includes an error prediction unit. The key disadvantage of this design is the broad critical path delay from input to output, despite the lower error rate and area overhead.

The BCSA approach [11] implements an n-bit approximate carry speculative adder using sub blocks, with carry prediction units performing parallel operations on these sub blocks. The BCSA method divides the n-bit adder into l-bit summation blocks, each subblock containing l-bits, to perform n-bit addition. Carry prediction and selection units are present in the sub adder. The carry from the previous block does not rely on the input of the next block to perform the parallel operation in each block.

Fig.1. Implementation of block based approximate adder with carry prediction unit.

As a result, there is no path between these blocks. Therefore, each sub adder will receive the input carry signal from the carry prediction unit of previous sub block. As shown in Fig.1, the i^{th}sub adder will receive the carry input from the $(i-1)^{th}$block.

The carry chain's length is determined by the carry prediction unit and selector unit. As a result of using these circuits, the critical path delay will be reduced since the carry input for the next sub block will be determined by these prediction blocks rather than the previous carry output. Since the carry input is dependent on the carry output of the previous block in the traditional method, the length of the carry chain is increased, so the above technique is used to truncate carry propagation to two blocks (in worst case).

The precision of the BSCA [11] approximate adder is higher since it will work as an exact adder in most situations. The logical expression below yields the carry input for the i^{th}adder block.

$$CO^i = \overline{sel^i}.C^i_{ADD} + sel^i.C^i_{prdt} \qquad (1)$$

Here CO^i denotes the carry output from the i^{th} block. The output of the selector unit is sel, as seen in the logical expression above. $C^i_{ADD}$is used to represent the sub-carry adder's output. Where C^i_{prdt}denotes the expected carry value.

$$sel^i = K^{i+1}_k + G^i_{l-1} \qquad (2)$$

$$C^i_{prdt}=G^i_{l-1} \qquad (3)$$

$$C^i_{ADD}=P^i_{l-1}G^i_{l-2} + P^i_{l-1}P^i_{l-2}G^i_{l-3} + \cdots + \prod_{k=1}^{l-1} P^i_k G^i_0 \qquad (4)$$

Where $C^i_{ADD}$and C^i_{prdt} are the carry outputs provided by each sub-block selected by sel^i. Certain cases are implemented using the kill bit (k), and the generate bit (G) value produced by the NOR operation on the input data are used. In certain instances, there is a higher error rate, which is partially restored by the Error Recovery Unit (ERU).

As shown in Fig.2, a block-based approximate adder is implemented. Where the n bit adder is divided into l-sub

blocks, each

Fig.2. Structure of approximate adder with error recovery unit

of which contains sub adders that are implemented using a ripple carry adder. An error recovery block is used to increase the sub adder's accuracy while lowering the estimated adder's error rate.

The work in [12] proposes an approximate ripple carry adder in which the n-bit adder is split into segments with no relation between them. The first and last full adders in each sub block are replaced by a designed full adder cells, which improves accuracy while increasing critical path delay due to the ripple carry adder in these blocks.

The work in [13] proposed an Approximate reverse carry propagation Full Adder (RCPFA) in this paper Carry propagates from Most significant bits (MSBs) to Least significant bits (LSBs). In terms of delay variation Approximate reverse carry propagation Full Adder offered more stability.

COMPARATIVE EVALUATION

In this section will compare various state of art Approximate Adders with Carry look ahead adder.Device utilization summary, Performance parameter summaryand Error metrics evaluationare discussed. Analysis done on four different state-of-art Approximate adders such as Speculative adder, Parallel Prefix adder, Approximate Adder Cells and Reverse carry Propagate adder.

I. Carry Look a Head Adder (CLA)

Carry-Look ahead Adder is presented in [9], [16]The principles of generating and propagating carries are used in carry-lookahead logic.The carry look ahead adder determines the carry in advance based primarily on input data signal. As a result, the propagation delay of the carry is reduced. The process of the carry look ahead adder can be understood by considering the complete adder's Boolean expression. Generate and Propagate Functions for full adder are given in the below (7), (8).

II. Block Based Carry Speculative Adder (BCSA)

In [11] block-based carry speculative adder is proposed in which adder was divided into non overlapping sum blocks of length l-bits, carry from the previous block is truncated to two blocks. The carry output of each block is estimated using the block's input operands and those of the following block.Here analysis was done on 32bit adder and block size of 8-bit. BCSA was implemented in two cases with Error recovery unit (ERU) and without error recovery here we have done analysis with ERU.

III. Approximate Adder (AA)

This work was proposed in [14]. In this method, approximate adder circuit is implemented by selecting approximate carry output. For the analysis purpose Approximate Adder cell was cascaded to 8-bit and placing ERU at each 8-bit block to form 32-bit adder similarly as in [9-12] methods exact full adder are cascaded to implement the adder function. Here wereplace the exact full adders by using approximate full adder in order to reduce the hardware complexity. The below Fig.3 shows the design of approximate adder diagram.

The approximate full adder correctly performs the sum operation, but it produces approximate carry output, which can be recovered using AND and OR gates as shown in Fig.3. The following is the logical expression (5)- (6).

$$S^i_{[l-1:0]} = a^i_{[l-1:0]} \oplus b^i_{[l-1:0]} \oplus C^{i-1} \quad (5)$$

$$C^i = a^i_{[l-1:0]} + (b^i_{[l-1:0]}.C^{i-1}) \quad (6)$$

Approximate adders's accuracy is dependent on the size of the approximate block as well as the carry predict and select logic. Carry propagation is restricted to block itself, which reduces delay. So the maximum adder delay is the same as the delay of the approximate adder block. In AA, approximate adder cell is used, which violates carry

output only in two cases: when a ='0', b ='1' &c_{in}= '0'and a ='1', b = '0' &c_{in}= '0'. Recovery logic was placed in each approximate adder cell for this violation, ensuring that the proposed approach achieves-high speed and tolerable error rate.

Carry propagation is performed at the block level using select logic, which is the sum of the kill bit (k) and the generate bit (g) (G). As shown in Fig.3, K and G values are obtained from the NOR of input data.

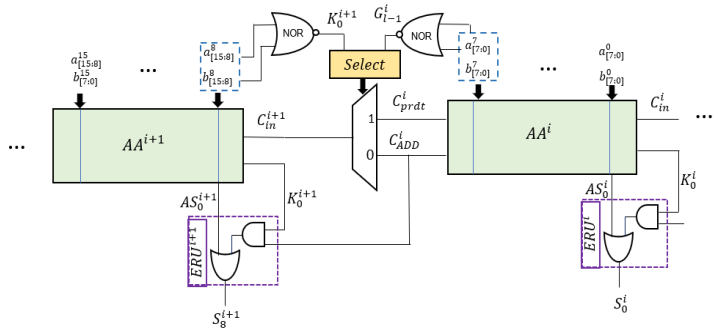

Fig.3 Approximate adder

IV. Approximate Han-Carlson Adder (AHA)

Approximate Han-Carlson Adder (AHA) is discussed in [15].The Han Carlson adder, which is one of the fastest parallel prefix adders, is used in this work. As a result, the Han-Carlson adder significantly decreases the critical path latency, resulting in increased processing speed and accuracy. The addition operation is done in three stages in a parallel prefix adder.

1.Pre-processing stage
2.Carry generation stage
3.Post processing stage

1.Pre-processing stage

Propagate p_i and generate G_i values are implemented in the same way as the carry look ahead adder at this point. For each input, these propagate and generate values are determined and sent to the next level.

$$p_i = A_i \oplus B_i \quad (7)$$

$$G_i = A_i B_i \quad (8)$$

2. Carry generation stage

Parallel prefix adders have the same first and last stages. For each parallel prefix adder, the carry generation stage has a different structure. To obtain carry for this point, black cells and grey cells are used, as shown in Fig.5.

The Han-Carlson adder's carry generation stage is represented in Fig. 4.

Fig.4. Carry generation stage of Han Carlson Adder

Fig.5. (a)Black cell, (b) Internal logic of Black cell and (c)Grey cell, (d) Internal logic of Gray cell.

3. Post processing stage

This is the stage where adder's final sum is computed. Between the propagate values and previous stage carry values, the Ex-Or operation is performed.Below is the logical expression (9).

$$S_i = P_i \oplus C_{i-1} \tag{9}$$

In this way, the Han-Carlson adder is introduced and used instead of AA blocks in the AHA block. The critical path is reduced in the AHA block architecture, which improves the operation's speed and reduces delays significantly. The AHA block is 8 bits long, and the $C_{prdt}$and C_{ADD}variables are generated and routed as described in (1)-(4) select logic and predict logic was similar as AA.

Fig.6. Approximate carry speculative adder with AHA block

RESULTS AND DISCUSSION

The CLA, BCSA, RCPFA, AA and AHA are coded with Verilog HDL, Simulation and Synthes is done in Xilinx ISE 14.7. The description of device utilization, performance parameters, and error metrics measurement are discussed in this section. Table.1-4 shows the results of a comparison study of various FPGA families. As shown in Fig.7-8, area and delay reports are produced by averaging area and delay across different FPGA families, while error metrics are provided in Table.5.

I. Device utilization Summary:

CLA, BCSA, RCPFA, AA, and AHA are implementedin Xilinx FPGA board families such as the Vertex4, Vertex5, Artix7, and Kintex7. Differentiation is performed in four categories: the number of slices, the number of four input LUTs, the number of IOs, and the number of bonded IOs. Table 1 shows the contrast between them. Table.2shows the percentage change. As compared to CLA current BCSA, RCPFA, AA, and AHA, there is a 31 percent improvement in Hardware utilization.

Table.1.Device utilization summary of BCSA, ACSA & ACSHA

	Virtex4				Virtex6				Artix7				Kintex7			
	No of Slices	No of 4i/p LUTs	No of IOs	No of bonded IOBs	No of Slices	No of 4i/p LUTs	No of IOs	No of bonded IOBs	No of Slices	No of 4i/p LUTs	No of IOs	No of bonded IOBs	No of Slices	No of 4i/p LUTs	No of IOs	No of bonded IOBs
CLA	76	78	98	98	99	33	98	98	99	33	98	98	99	33	98	98
BSCA	66	115	98	98	75	33	98	98	75	33	98	98	75	33	98	98
RCPFA	40	63	97	97	59	29	97	97	60	23	97	97	60	23	97	97
AA	43	75	98	98	60	28	98	98	60	28	98	98	60	28	98	98
AHA	57	101	98	98	60	28	98	98	78	8	98	98	78	8	98	98

Table.2. Percentage improvement in device utilizationSummary of ACSA & ACSHA

	Virtex4				Virtex6				Artix7				Kintex7			
PARAMETER	Slices	4i/p LUTs	IOs	Bonded IOBs	Slices	4i/p LUTs	No of IOs	Bonded IOBs	Slices	4i/p LUTs	IOs	Bonded IOBs	Slices	4i/p LUTs	IOs	Bonded IOBs
BCSA	13%	-47%	0%	0%	24%	0%	0%	0%	0%	0%	0%	0%	24%	0%	0%	0%
RCPFA	47%	19%	1%	1%	40%	12%	1%	1%	30%	30%	1%	1%	39%	30%	1%	1%
AA	43%	4%	0%	0%	30%	15%	0%	0%	15%	15%	0%	0%	39%	15%	0%	0%
AHA	25%	-29%	0%	0%	30%	15%	0%	0%	76%	76%	0%	0%	21%	76%	0%	0%

Table.3. Performance parameters of BCSA, ACSA, ACSHA

	Virtex4				Virtex6				Artix7				Kintex7			
	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)
CLA	78	12.68	989	6.00	99	5.22	517	7.15	99	6.82	675	9.88	99	4.84	479	10.16
BCSA	115	12.68	1460	6.00	75	4.87	365	7.15	75	6.64	498	9.84	75	4.55	341	10.00
RCPFA	63	8.62	543	6.59	59	3.49	206	8.53	60	4.67	280	8.89	60	3.64	218	8.86
AA	75	7.85	589	5.83	60	3.19	191	6.76	60	4.36	262	9.28	60	2.96	178	9.14
AHA	101	7.98	806	5.9	60	3.19	191	6.76	60	4.22	253	9.72	60	2.87	172	9.56

II. Design Parameters Evaluation:

The results of the delay, Area, ADP and Simulation time of the 32-bit reviewed Approximate adders for different FPGA families are shown in Table.3. From the results of CLA, BCSA, RCPFA, AA, and AHA we can observe that RCPFA has minimum area and AHA has high speed that is shown in Fig.7 and Fig.8

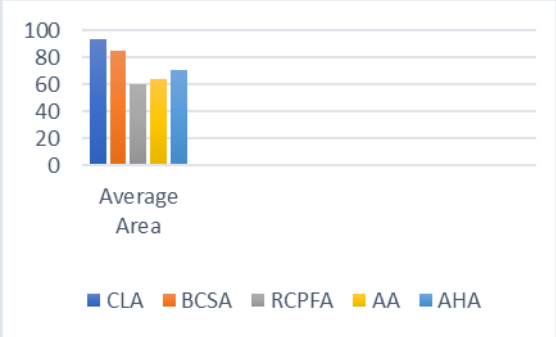

Fig.7. Area Utilization and Comparison of BCSA, ACSA, and ACSHA

The area report in terms of LUT's average from four distinct FPGA families is shown in Fig.7, and has significant changeas compared to CLA. As a result, when compared to the rest of the designs and the traditional CLA system, the RCPFA approach has a hardware complexity that is approximately 34% lower.

The delay report in nanoseconds between CLA, BCSA, RCPFA, AA, and AHA is shown in Fig.8. The time it takes to produce Generate and propagate values is greatly reduced as the gate count and effective length of the carry path are reduced. As a result. AA and AHA methods have less critical path latencywhen compared to CLA, BCSA, and RCPFA.

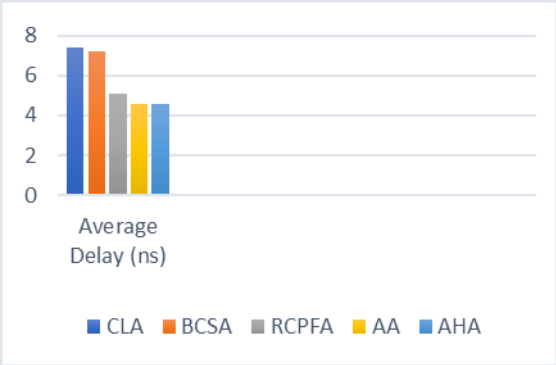

Fig.8. Delay Comparison of BCSA, ACSA, and ACSHA

Table 4 compares the output of reviewed approximate adders in terms of area, delay, and ADP. By comparing the all the five approximate adders,AA has 57% improvement and RCPFA has 55% improvement in ADP.AA gives higher percentage output gain in cost metrics.

Table.4. Percentage improvement in ACSA and ACSHA

	Virtex4				Virtex6				Artix7				Kintex7			
CLA	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)	Area	Delay(ns)	ADP	Simulation time(sec)
BCSA	-47%	0%	-47%	0%	24%	7%	29%	0%	24%	3%	26%	0.4%	24%	6%	29%	2%
RCPFA	19%	32%	45%	-10%	40%	33%	60%	-19%	39%	32%	59%	10%	39%	25%	55%	13%
AA	4%	38%	41%	3%	39%	39%	63%	6%	39%	36%	61%	6%	39%	39%	63%	10%
AHA	-29%	37%	19%	2%	39%	39%	63%	6%	39%	38%	63%	2%	39%	41%	64%	6%

V. Error Metrics Evaluation:

As previously mentioned, the Approximate adder's accuracy is determined by the select logic, predict logic, and block length. Each block in reviewed Approximate adders has different adder cells, and size varies from 1-bit to 8-bit are cascaded to form 8-bit block. For all the adders has same ERU that is placed at end of each block. Since the carry output was violating in two cases, we corrected it with ERU. We placed ERU at alternative bits in each block to get more benefit from Approximation.Similarly, parallel prefix adder blocks are used in AHA blocks, and this approximation is performed at the block level, as shown in Fig.6. Only select and predict logic determines accuracy in this case.Table.5 shows the error metrics obtained by applying random stimuli in 65K combinations for 16-bit and 32-bit. The results show that AA has a low delay while having bit error rate. BCSA and RCPFA has 1.7% and 1.3% relative error rate. AHA is largely reliable and has anacceptable cost function.

$$\mathrm{RE} = \left|\frac{s_i - s_i'}{s_i}\right| \qquad (10)$$

$$\mathrm{NMED} = \frac{1}{2^n}\sum_{i=0}^{n}\frac{s_i - s_i'}{2^n} \qquad (11)$$

$$\text{MRED} = \frac{1}{|n|}\sum_{i=0}^{n}\frac{|s_i - s_i'|}{s_i} \quad (12)$$

Table.5. Error metrics evaluation of CLA, BCSA, RCPFA, AA & AHA

	16			32		
	RE(%)	NMED	MRED	RE(%)	NMED	MRED
CLA	0.0	0.0	0.0	0.0	0.0	0.0
BCSA	1.70	0.0164	0.0185	0.20	0.0002	0.0003
RCPFA	1.3	0.0109	0.009	0.3	0.0021	0.0001
AA	5.60	0.05757	0.05019	4.30	0.0345	0.0375
AHA	0.01	0.0054	0.0	0.0	0.0	0.0

Where $s_i$is the exact result, $s_i'$is the approximate result, and n is the size of the adder. The mean relative error distance (MRED), normalised mean error distance (NMED), and relative error (RE) are determined from (10)-(12). The data is presented in a table. 5. The results show BCSA, RCPFA has <2percent error rate, AA has a 5.6 percent relative error (in 16-bit) that can be improved as a matter of designer interest by including ERU's at different bit positions, and AHA has a nearly negligible relative error.

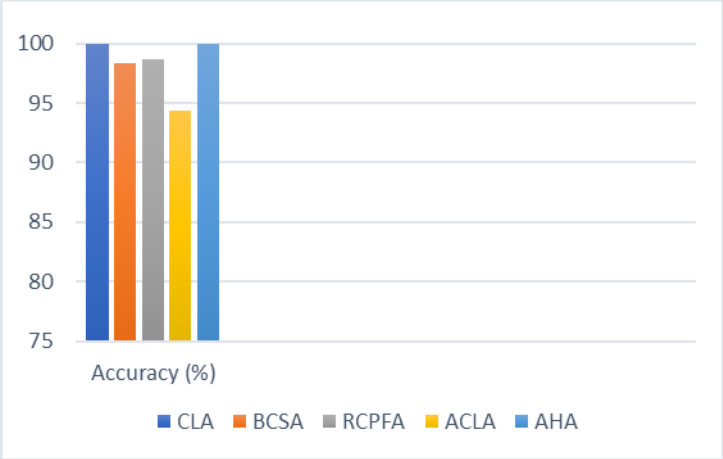

Fig.9. Error percentage in CLA. BCSA, RCPFA, ACLA, AHA

DISCUSSION AND CONCLUSION

This paper presents ancomparative evaluation of different types of Approximate adders. The selection of the Approximate adder depends upon the application that is used. Four approximate adders have been discussed, first adder that is BCSA have the less area compared with CLA, in the second adder is RCPFA which is one of the stable adder having optimal area, and we discussed AA, AHA that are having optimal speed and accuracy. Analysis is done in same input stimuli for all the reviewed approximate adders and error recovery block is adopted which will greatly reduce the overall error percentage. From the experimental results, we can conclude that RCPFA and AHA designs consists of optimal area and delay when compare to CLA, BCSA, AA approximate adders and conventional adders, In future this work can be implemented in various applications like MAC

(Multiply and Accumulation unit) and filters in order to estimate the performance of the proposed adders.DSP can be designed to get more advantage in cost function.

REFERENCES

1. Alexander Aponte-Moreno, Alejandro Moncada, Felipe Restrepo-Calle, Cesar Pedraza. "A Review of Approximate Computing Techniques towards Fault Mitigation in HW/SW Systems," 978-1-5386-1472-3/18/ 2018 IEEE DOI: 10.1109/LATW.2018.8347241.
2. C. M. Kirsch and H. Payer. "Incorrect systems: It's not the problem, it's the solution," in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, June 2012, pp. 913–917.
3. H. Jiang, C. Liu, L. Liu, F. Lombardi and J. Han. "A review, classification and comparative evaluation of approximate arithmetic circuits," ACM JETCAS, vol. 13, no. 4, Art. no. 60, 2017.
4. M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati and S. Mahlke. "SAGE: self-tuning approximation for graphics engines," In Proc. Micro, 2013, pp.13-24.
5. H. Esmaeilzadeh, A. Sampson, L. Ceze, and D.Burger. "Neural Acceleration for General-Purpose Approximate Programs," In Proc. of Micro, pp.105-115, 2012.
6. Weiquiang Liu, Fabrizio Lombardi, Michael Schultte. "Approximate Computing: From Circuits to Applications," Vol. 108, No. 12, December 2020, pp. 2103-2107.
7. B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst. "DVAFS: Trading computational accuracy for energy through dynamic-voltageaccuracy-frequency-scaling," in Design, Automation and Test in Europe (DATE), 2017 IEEE Conference, March 2017, pp. 488–493.
8. D. Mohapatra, G. Karakonstantis, and K. Roy. "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in Low Power Electronics and Design (ISLPED), 2009 ACM/IEEE International Symposium on, Aug. 2009, pp. 195–200.
9. O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. "RAP-CLA: A reconfigurable approximate carry look-ahead adder," IEEE TCAS-II, vol. 65, no. 8, pp. 1089–1093, 2018.
10. Y. Kim, Y. Zhang, and P. Li. "An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems," In Proc. ICCAD, 2013, pp. 130–137.
11. Farhad Ebrahimi-Azandaryani1, Omid Akbari, Mehdi Kamal, Ali Afzali-Kusha, MassoudPedram, "Block-based Carry Speculative Approximate Adder for Energy-Efficient Applications" IEEE Transactions on Circuits and Systems II,February 2019, DOI: 10.1109/TCSII.2019.2901060.
12. W. Xu, S. S. Sapatnekar, and J. Hu. "A Simple Yet Efficient Accuracy Configurable Adder Design," In Proc. ISLPED, 2017.
13. Masoud Pashaeifar, Mehdi Kamal, Ali Afzali-Kusha, MassoudPedram, "Approximate Reverse Carry Propagate Adder for
14. Energy-Efficient DSP Applications"IEEE transactions on very largescale integration (VLSI) systems, August 2018, DOI:10.1109/TVLSI.2018.2859939.
15. G. Anushaa, P. Deepa b, "Design of approximate adders and multipliers for error tolerant image processing,"Microprocessors and Microsystems Volume 72, February 2020, https://doi.org/10.1016/j.micpro.2019.102940.
16. Morgana Macedo, Leonardo Soares, Bianca Silveira, Claudio M. Diniz, Eduardo A. C. da Costa, "Exploring the Use of Parallel Prefix Adder Topologies into Approximate Adder Circuits" 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS) February 2017 DOI: 10.1109/ICECS.2017.8292078.
17. Neema Zacharias, Lalu V, "Study of Approximate Multiplier with Different Adders," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 10-12 Sept. 2020DOI: 10.1109/ICOSEC49089.2020.9215425.
18. Priyadarshini K.M., Ravindran R.S.E., Bhaskar P.R. (2019), 'A detailed scrutiny and reasoning on VLSI binary adder circuits and architectures', International Journal of Innovative Technology and Exploring Engineering, 8(7), PP.887-895.
19. Soumya N., Sai Kumar K., Raghava Rao K., Rooban S., Sampath Kuma R P., Santhosh Kumar G.N. (2019), '4-bit multiplier design using cmos gates in electric VLSI', International Journal of Recent Technology and Engineering, 8(2), PP.1172-1177.
20. Balaji B., Ajay Nagendra N., Radhamma E., Krishna Murthy A., Lakshmana Kumar M. (2019), 'Design of efficient 16 bit crc with optimized power and area in vlsi circuits', International Journal of Innovative Technology and Exploring Engineering, 8(8), PP.87-91.
21. Murali Krishna B., Madhumati G.L., Khan H. (2019), 'FPGA based pseudo random sequence generator using XOR/XNOR for communication cryptography and VLSI testing applications', International Journal of Innovative Technology and Exploring Engineering, 8(4), PP.485-494.

22. Santhosh C., Ravindran R.S.E., Vulchi U.B.P., Thumati V., Gufran M.S., Bhavana D., Cheerla S.V. (2019), 'Design and verification of half adder using look up table (LUT) in quantum dot cellular automata (QCA)', International Journal of Advanced Science and Technology, 28(16), PP.1804-1809.