Date of publication SEP-30, 2022, date of current version AUG-25, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.3.2696

High Speed Approximate Carry Speculative Adder in Error Tolerance Applications

AJAY KUMAR GOTTEM[1], ARUNMETHA SUNDARAMOORTHY[2], ARAVINDHAN ALAGARSAMY[3]

[1,2,3] Multi-core Architecture Computation (MAC) Lab, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, Guntur District 522 502, India

(e-mail: ajaykumar39751@gmail.com, sarunmetha@gmail.com, drarvindhan@kluniversity.in)

Corresponding Author: Aravindhan Alagarsamy (drarvindhan@kluniversity.in)

The authors thank DST - FIST for funding the lab facility for supporting this research under grant number SR/FST/ET-II/2019/450.

ABSTRACT Approximate adders were proposed as feasible solution in error-tolerant applications to provide a proper trade-off with accuracy over other circuit-based metrics like energy, area, and delay. State of art of approximate adders are shown in this work to improve the operational features significantly. To acquire a most benefits of approximation, in this paper approximation at lower echelons is presented. Two speculative adders are proposed, one with approximate adder cell and other with Parallel prefix Adder cell. Gate level implementation of proposed model are designed and implemented. The cost functions are compared against various FPGA standard architectures. Results of proposed approach indicate an average of 46% improvement in Area Delay Product (ADP) and compared with existing approximate adders.

KEYWORDS Approximate adder; Area delay product (ADP); Parallel prefix adder; Field programmable gate array.

I. INTRODUCTION

POWER optimization and speed enhancement are the main targets in the design of digital circuits. Approximate adder is used as a basic element in almost all arithmetic operations such as multiplication, subtraction, and division in digital circuits. These approximate adders gain significant attention in these recent years by the designers to implement various types of application where some error is tolerated [1-5]. Approximate adders gained importance because of high computation demand in the applications [6-8] such as machine learning, deep learning, image processing and digital signal processing (DSP) and wireless communication. In these applications we are introducing errors intentionally to get advantage of delay, area and power. Therefore, approximate adders have been implemented with significant loss since we don't require single golden result for those applications, however sufficient. Approximate computing was done at various abstraction levels like dynamic-voltage-accuracy-frequency-scaling in [9-11], computation at algorithmic level in and redesigning of circuits into approximate variants by inserting prediction logic that is especially suited for arithmetic adders [12-14].

The two approximate adders, one with approximate adder cell and other with Han-Carlson adder is implemented [15-17]. 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 cell (AAC) and Error Recovery Unit (ERU) added benefits to the proposed work that are evidently described in the further sections of this paper.

This paper is structured as follows. Section II explains about related works. Section III highlights the two proposed methods and its significance. Section IV experimental results

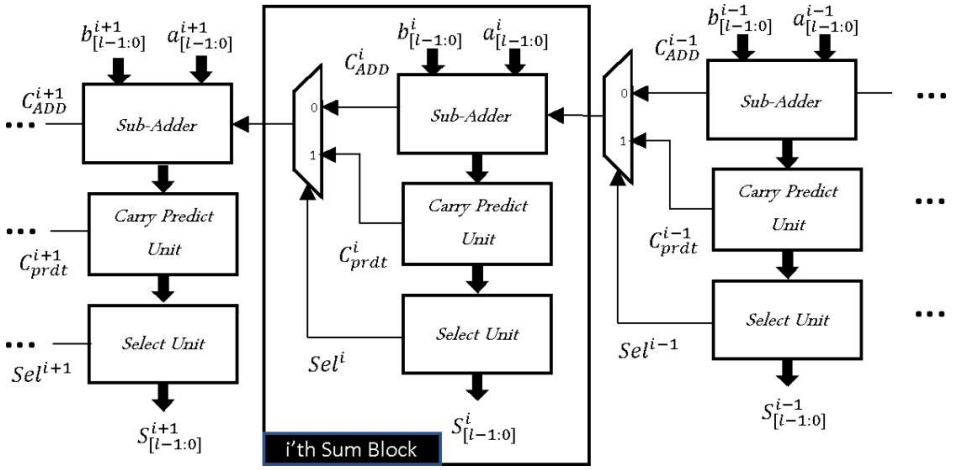

Figure 1. Implementation of block based approximate adder with carry prediction unit.

results are validated for the proposed model. Finally, conclusion and future work are discussed in section V.

II. RELATED WORKS

In this section we will briefly discuss about existing works, where state of art approximate adder is implemented. The work in [9] represents a reconfigurable approximate carry look ahead (CLA) adder implementation where it acts as both approximate as well as exact adder. It uses exact carry look ahead adder to implement the design and also multiplexer are used to select the accuracy configurability in order to act as exact adder and approximate adder. This way of implementing the design will cause more hardware complexity and more delay. In [10], an approximate carry skip adder (CSA) is proposed which is implemented by using exact carry skip adder. Where the n-bit adder is clustered into l-bit sub adders in which each sub adder gets the carry from the carry prediction block to reduce the overall critical path delay. An additional, error magnitude correct block is implemented in view of betterment to accuracy and decrease the error rate. Hence by using this additional block will increase the critical path delay and power consumption of the approximate adder design. So, this affects the cost function of the design.

The work in [11], a high accuracy block-based carry speculative adder (BCSA) is proposed, and it implemented by using carry propagate adder and consists of an error prediction unit. Although the error rate and area overhead are less, the main drawback of this design is large critical path delay from input to output. An approximate ripple carry adder (RCA) is proposed in [12], in which n-bit adder is clustered into sub-blocks and there is no connection between the sub blocks. In every sub block the first and last full adders are replaced by the modified full adder to where accuracy is more with increase in critical path delay because of ripple carry adder in these blocks. In BCSA approach [11], n-bit approximate carry speculative adder is implemented by using sub blocks in which parallel operation of these sub blocks is implemented by using carry prediction unit. To perform n-bit addition in the BCSA method; the n-bit adder is clustered into l-bit summation blocks in which each sub blocks consists of l-bits. In the sub adder, carry prediction unit and selection unit is present. In each block to perform the parallel operation, the carry from the previous block does not relay with the input of the next block. Hence there is no path exist between these blocks. Hence each sub adder will get the input carry signal from its previous carry prediction unit. It indicates the i^{th} sub adder will get the carry input from the $(i-1)^{th}$ block as shown in Fig.1.

The carry chain length depends upon the carry prediction unit and selector unit [18]. Hence by using these circuits will trims down the critical path delay, due to the carry input for the next sub block will depend upon these prediction blocks not from the previous carry output [19-21]. In conventional method, the carry chain length is more because the carry input is dependent upon the carry output of the previous block so above method is used to truncate carry to two blocks (in worst case). In BSCA [11] approximate adder, the accuracy is more because most of the cases it will behave as an exact adder. The carry input for the i^{th} adder block is obtained from the below logical expression.

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

where CO^i is carry output from the i^{th} block. From the above logical expression *sel* is the output from the selector unit. The carry output from the sub-adder is denoted by using C^i_{ADD}. Where predicted carry value is consider as C^i_{prdt}.

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

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

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

where C^i_{ADD} and C^i_{prdt} are carry outputs generated from each sub-block that is selected by sel^i. Certain cases are implemented based on generate values and kill bit that we get from NOR operation on input data. In some of the cases have higher error rate that is some extent recovered by Error Recovery Unit (ERU). The block based approximate adder is proposed as shown in Fig.2. Where n bit adder is segmented with l-sub blocks in which each sub block consists of sub adders which are implemented by using ripple carry adder. An

error recovery block is used to improve the accuracy of the sub adder while decreasing the error rate of the approximate adder.

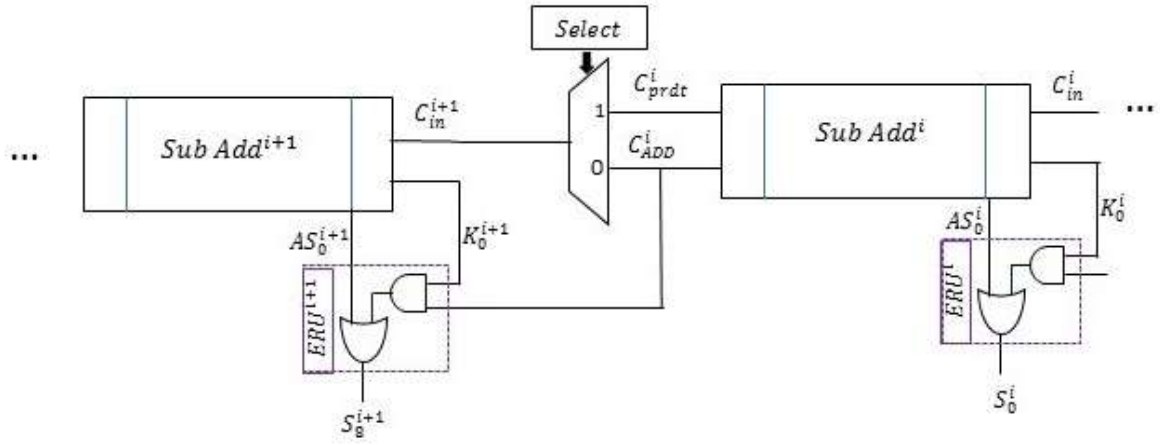

Figure 2. Structure of approximate adder with error recovery unit

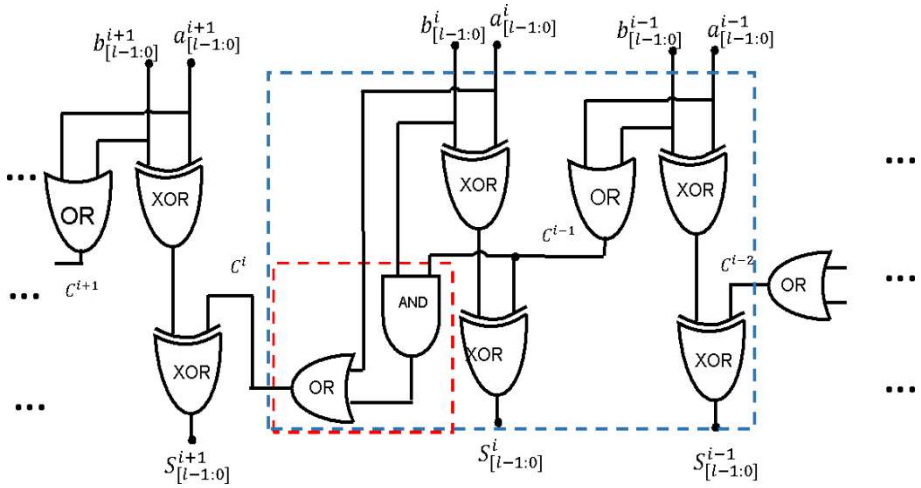

Figure 3. Approximate adder cell

III. PROPOSED INTERNAL ARCHITECTURE

In this section will implement a high-speed approximate carry speculative adder is proposed. Fig 3 represents the design of approximate full adder cell. To improve the speed of the operation and decrease the hardware complexity two methods are proposed. One method is implemented by using approximate full adder cell that is Approximate Carry Speculative Adder (ACSA) as shown Fig.4. This method will decrease the delay and area parameters with slightly increasing in error rate. Second approach, instead of approximate adder cell, Han-Carlson adder is placed in ACSA block called Approximate Carry Speculative Han-Carlson Adder (ACSHA) as shown in Fig.7. adder is going to offer betterment in the critical path delay which will improve the speed of the operation.

A. APPROXIMATE CARRY SPECULATIVE ADDER (ACSA)

In this method, approximate adder circuit is designed, in in existing [9-12] methods exact full adder is cascaded to implement the adder function. In the proposed ACSA, it replace the exact full adders by adopting approximate full adder to reduce the hardware complexity.

The proposed approximate full adder performs the sum operation correctly, but we get approximate carry output, that can be recovered with AND and OR gates as shown in the Fig. 3. The logical expression is shown in eqn. (5) and (6). From truth table C_{out} means exact carry output, C'_{out} means approximated carry output. Accuracy of ACSA depend on dimention of the approximate block and carry predict & select logic. Delay is reduced since carry propagation is restricted to block itself. So, the maximum delay of entire adder is same as delay of approximate adder block. In ACSA, approximate adder cell is used that is violating carry output only in two cases, when a =’0’, b =’1’ & c_{in} = ’0’and a =’1’, b = ’0’ & c_{in} = ’0’ for this violation, recovery logic was placed in each approximate adder cell, so that the proposed approach achieves-high speed and tolerable error rate. For an 8-bit ACSA there consist of four approximate adder cells. At the block level carry propagation is done through select logic which is the sum of kill bit (k) and generate bit (G). Where K and G values receives from NOR of input data as in Fig.4.

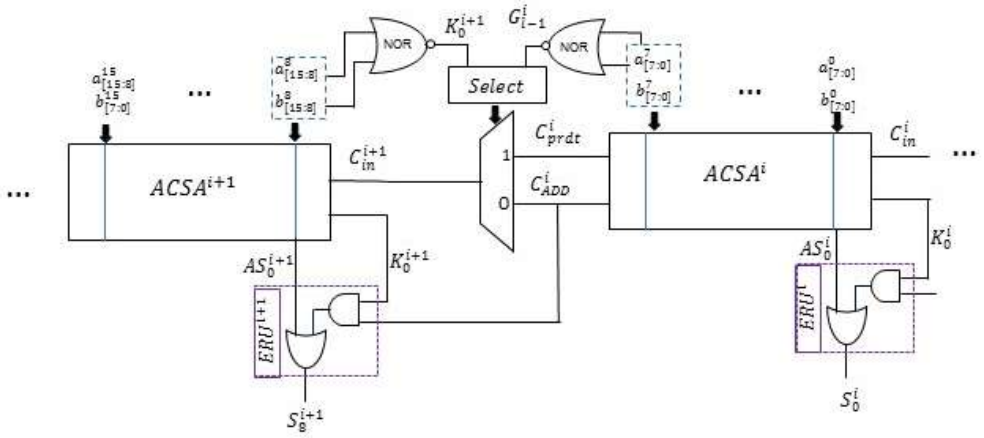

Figure 4. Approximate carry speculative adder

Table 1. Truth table for ACSA 1-bit

a	b	c_{in}	sum	c_{out}	c'_{out}
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

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

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

B. Approximate Carry Speculative Han-Carlson Adder (ACSHA)

ACSA block implemented using parallel prefix adder. In ACSHA method Han Carlson adder is used which is one among the fastest parallel prefix adders. Hence Han-Carlson adder greatly trims down the critical path latency with the betterment of speed of operation and better accuracy. In parallel prefix adder, the addition operation is performed by using three stages:

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

B.1 Pre-processing stage

In this stage the propagate p_i and generate G_i values are implemented as same as carry look ahead adder. These propagate and generate values are estimated for each input and sends to next stage.

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

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

B.2 Carry generation stage

First and last stage is same for all parallel prefix adders. The structure of carry generation stage is different for different parallel prefix adder. The carry generation stage of Han-Carlson adder is shown Fig. 5. The stage is implemented by using black cells and grey cells to obtain carry for this stage which is shown in Fig. 6.

B.3 Post processing stage

Final sum of the adder is calculated in this stage. The Ex-Or operation is performed between the propagate values and previous stage carry values. The logical expression is shown in eqn. (9).

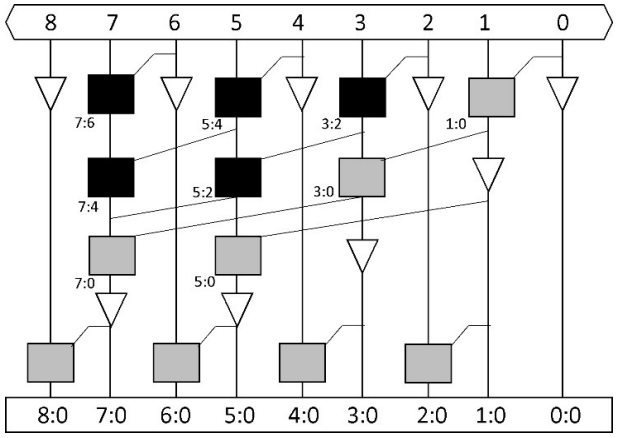

Figure 5. Carry generation stage of Han Carlson Adder

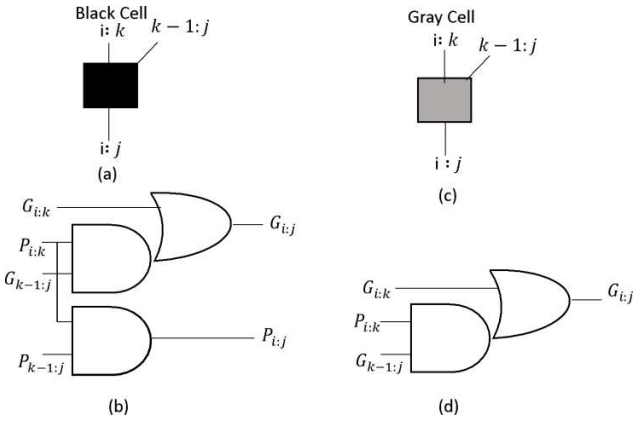

Figure 6. (a)Black cell, (b) Internal logic of Black cell and (c)Grey cell, (d) Internal logic of Gray cell.

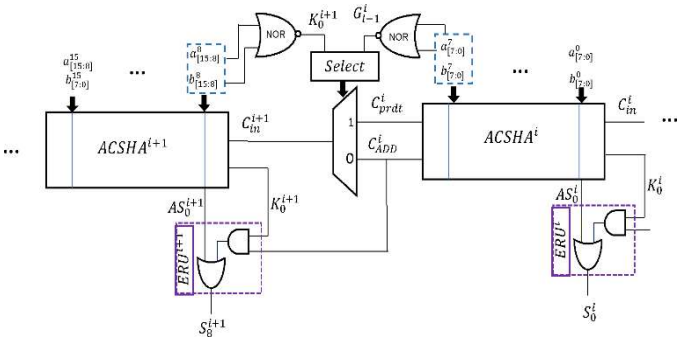

Figure 7. Approximate carry speculative adder with ACSHA block

In this manner the Han carlson adder is implemented and used in ACSHA block in place of ACSA blocks. The critical path is less in ACSHA block design which will improve the speed of the operation and has significant delay improvement. ACSHA block is of 8-bit length, C_{prdt} and C_{ADD} are generated and routed as mentioned in (1)-(4) select logic and predict logic was similar to ACSA.

IV. RESULTS AND DISCUSSIONS

The ACSA and ACSHA are coded with Verilog HDL, Simulation and Synthes is done in Xilinx ISE 14.7. In this section device utilization summary, performance parameters and error metrics evaluation is presented. Comparison analysis with different FPGA families is done as shown in Table.2-5. Simulation results shown in the Fig.8 and Fig.9. Area and delay reports are obtained through the average of area and delay in different families of FPGA as shown in the Fig.10-11 and error metrics is presented in Table.5.

A. DEVICE UTILIZATION SUMMARY

ACSA and ACSHA is implemented in Vertex4, Vertex5, Artix7 and Kintex7 of Xilinx FPGA families. Differentiation is done in four categories that is, No of slices, No of 4 input LUT's, No of IO's and bonded IO's. Percentage improvement mentioned in Table.3. On an average of 21% improvement is observed in ACSA and ACSHA.

B. DESIGN PARAMETER EVALUATION

According to the implementation, the outcome of the latency and execution time of the 32-bit studied adders are presented in Fig.8 and Fig.9, respectively. a[31:0], b[31:0], c_{in} and y[32:0] are input and output values represented in decimal, remaining are internally generated signals. The above Fig.10 shows the area report of in terms of LUT's average from four different FPGA families. and has change compared with BCSA. Hence ACSA method consists of less hardware complexity nearly 35% when compare to the existing [11] design.

Fig.11 shows the delay report in terms of Nanoseconds between BCSA [11], ACSA and ACSHA. As we are reducing gate count and effective length of the carry path time taken to generate and propagation is very less. Hence ACSA and ACSHA method consists of less critical path delay when compared to BCSA.

C. ERROR METRICS EVALUATION

As previously stated, the precision of the Approximate adder

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

depends upon the select logic, predict logic and length of the block, in ACSA each block has four approximate adder cells, each cell is of size 2-bit. It consists of XOR gates for sum and OR gates for carry since carry output was violating in two cases so that is corrected with ERU, we are placing ERU at alternative bits to get more advantage from Approximation as shown in Fig.3. Similarly, in ACSHA blocks are replaced

Figure 8. Simulation waveform of ACSA

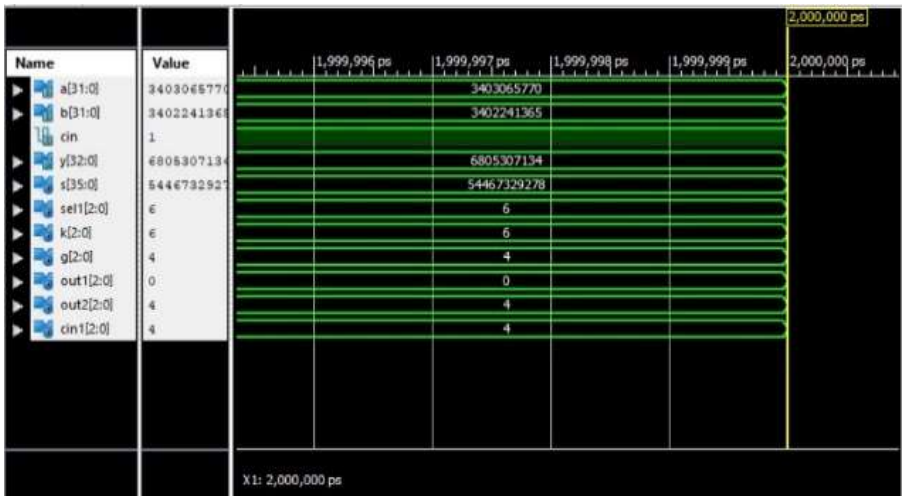

Figure 9. Simulation waveform of ACSHA

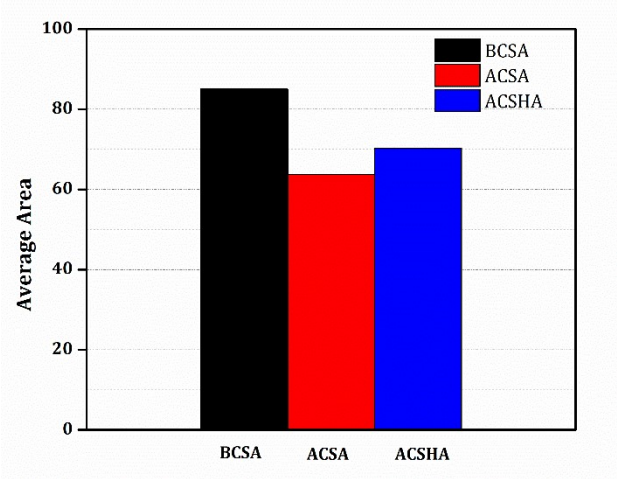

Figure 10. Area Utilization and Comparison of BCSA, ACSA, and ACSHA

with parallel prefix adder block, in this approximation is done at block level as shown in Fig.7.

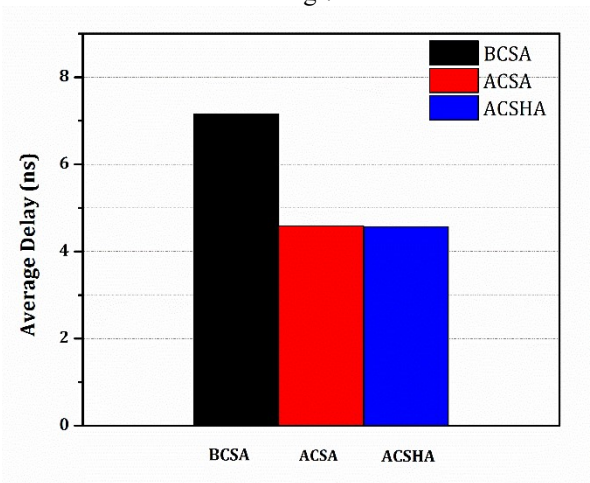

Fig.11. Delay Comparison of BCSA, ACSA, and ACSHA

Here accuracy depends upon only select and predict logic. Error metrics for both ACSA and ACSHA were obtained by applying random stimuli in 65K combinations for 16-bit and 32-bit as presented in Table.6. Results show that ACSA has low delay while having bit error, to improve the accuracy ERU can be placed at different points in approximate adder cell as designer's interest (Fig.3) and ACSHA is mostly accurate with relatively same cost function as ACSA.

$$\mathrm{RE} = \left|\frac{s_i - s_i'}{s_i}\right| \tag{10}$$

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

$$\mathrm{MRED} = \frac{1}{|n|}\sum_{i=0}^{n} \frac{|s_i - s_i'|}{s_i} \tag{12}$$

Where s_i' is exact result, s_i^''is approximate result, n is adder bit length.

Table 2 Device utilization summary of BCSA, ACS & ACSHA

Architec-ture Type	Virtex4				Virtex6				Artix7				Kintex7			
	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs
BSCA [11]	66	115	98	98	75	33	98	98	75	33	98	98	75	33	98	98
ACSA	43	75	98	98	60	28	98	98	60	28	98	98	60	28	98	98
ACSHA	57	101	98	98	60	28	98	98	78	8	98	98	78	8	98	98

Table 3 Percentage improvement in device utilization summary of ACSA & ACSHA

Architec-ture Type	Virtex4				Virtex6				Artix7				Kintex7			
	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs	# Slices	# 4 input LUT	# IOs	# Bonded IOBs
ACSA	35	35	0	0	20	15	0	0	20	15	0	0	20	15	0	0
ACSHA	14	12	0	0	20	15	0	0	20	75	0	0	20	75	0	0

Table 4 Performance parameters of BCSA, ACSA, ACSHA

Architec-ture Type	Virtex4				Virtex6				Artix7				Kintex7			
	Area	Delay	ADP	Time (s)	Area	Delay	ADP	Time (s)	Area	Delay	ADP	Time (s)	Area	Delay	ADP	Time (s)
BSCA [11]	115	12.68	1460	6.00	75	4.87	365	7.15	75	6.64	498	9.84	75	4.55	341	10
ACSA	75	7.85	589	5.83	60	3.19	191	6.76	60	4.36	262	9.28	60	2.96	178	9.14
ACSHA	101	7.98	806	5.9	60	3.19	191	6.76	60	4.22	253	9.72	60	2.87	172	9.56

Table 5 Percentage improvement in ACSA and ACSHA

Parameter		Virtex4	Virtex6	Artix7	Kintex7
		BCSA			
ACSA	**Area**	35%	20%	20%	20%
	Delay	38%	35%	34%	35%
	ADP	60%	48%	47%	48%
	Simulation time	10%	6%	6%	9%
ACSHA	**Area**	12%	20%	20%	20%
	Delay	37%	35%	37%	35%
	ADP	45%	48%	49%	50%
	Simulation time	1.2%	6%	10%	4%

Relative error (RE) normalized mean error distance (NMED) and mean relative error distance (MRED) is calculated from eqn. (10)-(12). Presented in Table.5. Results show that in ACSA 5.6% relative error (in 16-bit) that can be improved as of designer's interest as mentioned above and in ACSHA almost negligible.

Table 6 Error metrics evaluation of BCSA, ACSA & ACSHA

Logic	16 - Bit			32 - Bit		
	RE	NMED	MRED	RE	NMED	MRED
BCSA	1.7	0.02	0.02	0.2	~0	~0
ACSA	5.6	0.06	0.05	4.3	0.03	0.04
ACSH	0.0	~0	~0	~0	~0	~0

V. CONCLUSION

In this paper, a high-speed approximate carry speculative adder is proposed. To limit the critical path and improve the execution time, two approaches have been proposed, first approach to replace full adder by using approximate adder cells that has improved ADP and in second approach approximate adder cells are replaced by parallel prefix adder which greatly trims down the critical path latency and improved Accuracy. In this design, to improvise the proposed approximate adder accuracy, an error recovery block is adopted. From the experimental results, we can conclude that proposed designs consist of optimal area and delay when compare to existing approximate adders, further it can be extended to design n-bit parallel adders and accuracy tuned approximate Adders.

References

[1] A. Aponte-Moreno, A. Moncada, F. Restrepo-Calle and C. Pedraza, "A review of approximate computing techniques towards fault mitigation in HW/SW systems," *Proceedings of the 2018 IEEE 19th Latin-American Test Symposium (LATS)*, 2018, pp. 1-6. https://doi.org/10.1109/LATW.2018.8347241.

[2] C. M. Kirsch and H. Payer, "Incorrect systems: It's not the problem, it's the solution," *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2012, pp. 913–917. https://doi.org/10.1145/2228360.2228523.

[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. https://doi.org/10.1145/3094124.

[4] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati and S. Mahlke, "SAGE: self-tuning approximation for graphics engines," Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, 2013, pp. 13-24. https://doi.org/10.1145/2540708.2540711.

[5] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural Acceleration for general-purpose approximate programs," *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture MICRO-45*, 20121, pp. 105-115. https://doi.org/10.1109/MICRO.2012.48.

[6] W. Liu, F. Lombardi, M. Schultte, "Approximate computing: From circuits to applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2103-2107, 2020. https://doi.org/10.1109/JPROC.2020.3033361.

[7] K. M. Priyadarshini, R. S. E. Ravindran, P. R. Bhaskar, "A detailed scrutiny and reasoning on VLSI binary adder circuits and architectures," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, issue 7, pp. 887-895, 2019.

[8] N. Soumya, K. Sai Kumar, K. Raghava Rao, S. Rooban, R P. Sampath Kuma, G. N. Santhosh Kumar, "4-bit multiplier design using CMOS gates in electric VLSI," *International Journal of Recent Technology and Engineering*, vol. 8, issue 2, pp. 1172-1177, 2019. https://doi.org/10.35940/ijrte.B1742.078219.

[9] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst. "DVAFS: Trading computational accuracy for energy through dynamic-voltageaccuracy-frequency-scaling," *Proceedings of the 2017 IEEE Conference on Design, Automation and Test in Europe (DATE)*, March 2017, pp. 488–493. https://doi.org/10.23919/DATE.2017.7927038.

[10] B. Balaji, N. Ajay Nagendra, E. Radhamma, A. Krishna Murthy, M. Lakshmana Kumar, "Design of efficient 16 bit CRC with optimized power and area in VLSI circuits," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, issue 8, pp. 87-91, 2019.

[11] B. Murali Krishna, G. L. Madhumati, H. Khan, "FPGA based pseudo random sequence generator using XOR/XNOR for communication cryptography and VLSI testing applications," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, issue 4, pp. 485-494, 2019.

[12] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), Aug. 2009, pp. 195–200. https://doi.org/10.1145/1594233.1594282.

[13] C. Santhosh, R. S. E. Ravindran, U. B. P. Vulchi, V. Thumati, M. S. Gufran, D. Bhavana, S. V. Cheerla, "Design and verification of half dder using look up table (LUT) in quantum dot cellular automata (QCA)," *International Journal of Advanced Science and Technology*, vol. 28, issue 16, pp. 1804-1809, 2016.

[14] 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. https://doi.org/10.1109/TCSII.2016.2633307.

[15] Y. Kim, Y. Zhang, and P. Li, "An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems," Proceedings of the International Conference on Computer Aided Design ICCAD, 2013, pp. 130–137. https://doi.org/10.1109/ICCAD.2013.6691108.

[16] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, "Block-based carry speculative approximate adder for energy-efficient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 137-141, 2020. https://doi.org/10.1109/TCSII.2019.2901060.

[17] W. Xu, S. S. Sapatnekar, and J. Hu, "A simple yet efficient accuracy configurable adder design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol 26, issue 6, pp. 1112-1125, 2018. https://doi.org/10.1109/TVLSI.2018.2803081.

[18] T. Han and D. A. Carlson, "Fast area efficient VLSI adders," *Proceedings of the 8th IEEE Symposium on Computer Arithmetic*, Como, Italy, 1987, pp. 49-56. https://doi.org/10.1109/ARITH.1987.6158699.

[19] K. Vitoroulis and A. J. Al-Khalili, "Performance of parallel prefix adders implemented with FPGA technology," Proceedings of the 2007 IEEE Northeast Workshop on Circuits and Systems, 2007, pp. 498-501. https://doi.org/10.1109/NEWCAS.2007.4487969.

[20] B. Ramkumar and H. M Kittur, "Low-power and area-efficient carry select adder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 371-375, 2012. https://doi.org/10.1109/TVLSI.2010.2101621.

[21] F. Liu, F. Fereydouni Forouzandeh, O. A. Mohamed, G. Chen, X. Song and Q. Tan, "A comparative study of parallel prefix adders in FPGA implementation of EAC," Proceedings of the *2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, 2009, pp. 281-286. https://doi.org/10.1109/DSD.2009.135.

***Ajay Kumar Gottem** was born in Amaravathi Village, Andhra Pradesh, INDIA in 1997. He received the B.tech degree in electronics and communication engineering from RVR and JC College of Engineering, Andhra Pradesh, India, in 2019. He is currently pursuing the M.tech degree in Very Large Scale Integration (VLSI) at KL University, Andhra Pradesh, India.*

***ARUNMETHA SUNDARAMOORTHY** is an Associate Professor in Department of ECE, KLEF (Deemed to be University), Guntur- Andhra Pradesh. He completed SERB-National Post-Doctoral Fellow (N-PDF) in Anna University during the year 2016-2018. He was completed Ph. D in the year 2016 from Anna University with CSIR-SRF fellowship. He focused his research and development work on Nano materials for energy and environmental application. He has to his credit more than 23 research papers in reputed International journals, 6 papers in conference proceedings and 2 patents are applied. Under his able guidance, 6 M. Tech., students have completed their M. Tech project work. Now, he his guiding 3 Ph. D scholars, His research interests include Embedded Systems, Internet of Things (IoTs) enabling technologies, and smart cities enabling services.*

ARAVINDHAN ALAGARSAMY *received his B.E. in Electrical and Electronics Engineering from Madurai Kamaraj University, Madurai, India in 2003.He awarded the M. Tech., Degree in VLSI Design from Kalasalingam Academy of Research and Education, Tamilnadu India in 2009 and the Ph. D., Degree from National Institute of Technology, Tiruchirappalli in the area of Networks-on-Chip. Currently, he is in a position of Associate Professor, Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur Dist., AP, India. He had industrial experience as Design Engineer in Electronic Design Automation Industry. His research interest includes Network on Chip, Optimization, ASIC and FPGA and Soft Computing.*