A Novel and Efficient Left-to-Right Binary Adder Architecture for reduced Area and Power Metrics in VLSI Design

M. G. Anuradha
Department of Electronics and Communication, JSS Academy of Technical Education, (affiliated to Visvesvaraya Technological University, Belagavi), Dr. Vishnuvardhan road, Bengaluru-560060, India
anuradhamg@jssateb.ac.in (corresponding author)

R. Arun Kumar
Ohana Community School, Bengaluru, Karnataka, India
arun1979arun@gmail.com

Received: 5 December 2024 | Revised: 9 January 2025 | Accepted: 11 January 2025

 DOI: https://doi.org/10.48084/etasr.9840

ABSTRACT

Fast adders are utilized to cater for computationally intensive operations, and until recently, researchers have focused on optimizing the logic used in carry propagation. In the present study, a new methodology of implementation deploying the left-to-right Vedic addition method is proposed. The proposed methodology and the developed architecture add the number from the Most Significant Bit (MSB), where the carry generation is minimal, and reduce the complexity involved in the binary addition to optimize the area delay and power delay of the binary adder. The proposed left-to-right adder has been implemented for bit sizes of 8, 16, and 32. The synthesis results show that it outperforms existing adder techniques in terms of power-delay and area-delay products. The results indicate that the 32-bit left-to-right adder achieves an average reduction of 8% in the power-delay product and 23% in the area-delay product compared to current fast adders.

Keywords-fast adders; VLSI; left-to-right adder; Vedic mathematics; Verilog

I. INTRODUCTION

The adder is the basic arithmetic component in any computing machine. When designing a modern fast processor or when analyzing any multimedia content using any machine learning or pattern recognition algorithm, computations need to be performed at a higher rate. The algorithms used to analyze the multimedia content are computationally intensive, and most are iterative in nature. In most computing systems, the adder determines the critical path, and thus has a significant impact on the overall speed of the processing engine. In addition, due to their high switching activity, adders tend to become thermal hotspots in high-performance computing engines. Improving the performance of the basic adder element would lead to an overall improvement in the efficiency of the computing system. As a result, the adder design has become a focal point for researchers, leading to the development of various adder architectures tailored to meet the specific requirements of the computing systems, such as low power consumption and high speed.

In order to meet the different requirements of computing systems, various adders have been designed in the literature. The Ripple Carry Adder (RCA) is the basic binary adder, which is designed using multiple full adders. The main issue with using an RCA is its time complexity. In an RCA, the last full adder must wait for the carry to be generated, making the carry chain a significant problem in the binary adder circuits. As the number of input bits increases, the carry chain grows, causing propagation delay, and therefore speed reduction. This delay can be overcome by using carry select adders. The approach used in carry select adders is based on the principle that the carry generated at the end of each block can be either "1" or "0". To address this, the circuit is duplicated, with each full adder receiving both potential carry values. In terms of the area, an RCA is more efficient than a carry select adder. However, in terms of the propagation delay, the carry select adder is more efficient, with a delay that is half of that of the RCA. A review of the RCA and the carry select adder, focusing on the structural design and performance parameters is presented in [1]. Furthermore, the complexity of the RCA is minimized in [2] using quantum-dot cellular automata to reduce the power consumption and delay. A carry select adder with a multiplexer is also proposed in [3] to increase the computational speed in computationally intensive applications. In addition, authors in [4] designed a 32-bit carry select adder using a hybrid Carry-Lookahead Adder (CLA) to improve the computational efficiency. Authors in [5] proposed carry skip

adders to reduce the delay of RCAs. For even faster addition, parallel prefix adders are used because they effectively manage carry propagation. Unlike RCAs, which process each bit sequentially and suffer from carry propagation delay, parallel prefix adders compute multiple carry operations in parallel employing a more advanced method. This approach involves logic functions that determine whether the carry is generated or propagated through recursive equations. Various optimizations in the logic have led to different versions of parallel prefix adders, such as the Kogge-Stone adder [6], the Brent-Kung adder [2], Lee's adder [7, 8], and Jackson's adder [9]. Modulo adders are proposed in [10] to further optimize the speed of the parallel prefix adders and improve the computational efficiency in DSP applications [10]. Recently, the need for fast multimedia data processing in portable devices has prompted the adoption of approximate computing, where minor errors are acceptable due to human perceptual limitations [11]. As a result, various approximate adders have been developed to enable faster addition by reducing the length of the carry propagation chain. The performance tradeoffs of these specialized adders are analyzed in [12].

To further increase the speed of computation, various VLSI architectures have been introduced deploying the existing techniques for addition. Authors in [13] proposed an area efficient VLSI architecture for a three-operand binary adder and the authors in [14] optimized a three-operand binary adder architecture for area efficiency. Furthermore, authors in [15] optimized the architecture of a carry select adder to address the power-delay tradeoff and authors in [16] introduced block size optimization in carry select adders to improve efficiency. A square root carry select adder was proposed in [17] to increase the performance speed of the booth multiplier. Apart from changing the architectures of an adder, the performance of the adder is increased by introducing a new full adder cell [18, 19] to address the need for power performance in portable devices. Thus, the performance of computing systems is optimized by either new adder algorithms or new adder architectures.

This paper introduces a new binary adder architecture inspired by the principles of Vedic mathematics and mental addition techniques. The proposed method carries out the addition from the left, reducing the number of carry generations, and performs addition using only half adders and XOR gates. Unlike the existing adder designs that focus primarily on speeding up carry propagation and optimizing carry logic, the proposed architecture breaks the conventional rule of starting the addition from the Least Significant Bit (LSB) by starting from the MSB. The introduced architecture has been implemented with 8-, 16-, and 32-bit adders, and its performance is compared with other adders, such as the CLA, the Carry-Save Adder (CSA), and the Kogge-Stone parallel prefix adder. The results demonstrate that the proposed design and implementation outperform these traditional adders in terms of power efficiency and delay. In summary, the key contributions of the present article are:

- It proposes an innovative adder architecture based on a left-to-right mental math technique that minimizes the carry generation.
- The proposed addition method provides faster computation with lower power consumption, making it suitable for applications requiring fast, low-power processing.

II. METHODOLOGY

The addition process is taught to us by adding the numbers from one's place and if the addition results in carry, then the carry should be added to the ten's place and the addition continues in a similar manner. This type of addition is also carried in binary addition, causing a delay in computing the result due to carry propagation. As described in the previous section, many adder techniques have been proposed in the literature to compute the results faster and to avoid carry propagation. In Vedic and mental mathematics, the addition is carried out from the most significant digit to speed up the addition and avoid the carry propagation. The method is explained with an example. Consider the addition of two numbers 78 and 45. In the normal addition process, the addition starts by adding 8 and 5, which gives 13, and generates a carry of 1 on the tens, which must be added to 7 and 4 to give 123. Instead, the addition can be carried from the most significant digits 7 and 4, which gives 11. The addition continues by adding the two least significant digits to get 13. Now this answer is shifted to the right and added to 11 to get the same answer, 123, without the carry generation, as illustrated in Figure 1.

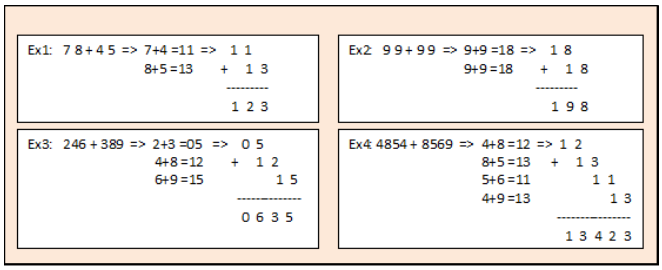

Fig. 1. Illustration of left-to-right addition.

As can be seen in Figure 1, carry is not generated in any of the additions. However, carry can be generated if and only if the sum of the higher-order digits is equal to 9 and the sum of the lower-order digits is greater than 10, for the decimal system. In general, carry can be generated if the sum of the higher-order digits is equal to 'base-1' and the sum of the lower-order digits is greater than the 'base', as depicted in Figure 2.

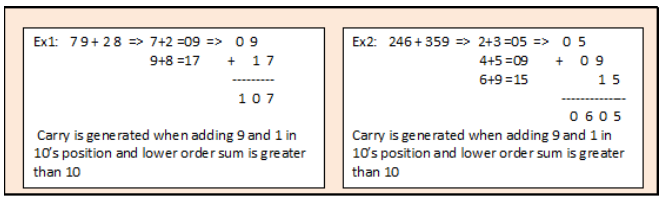

Fig. 2. Illustration of left-to-right addition when a carry is generated.

The same principle can be also applied to the binary addition, as shown in Figure 3. The addition process for binary numbers is the same as that for decimal numbers, and as seen,

the carry is generated if and only if the higher-order digit sum is equal to 1 and the lower-order digit sum is greater than or equal to the base which is 2 for the binary system. As observed, out of the 16 additions, carry is generated only for two cases when adding "01" with "11" and vice versa, as highlighted in orange in Figure 3.

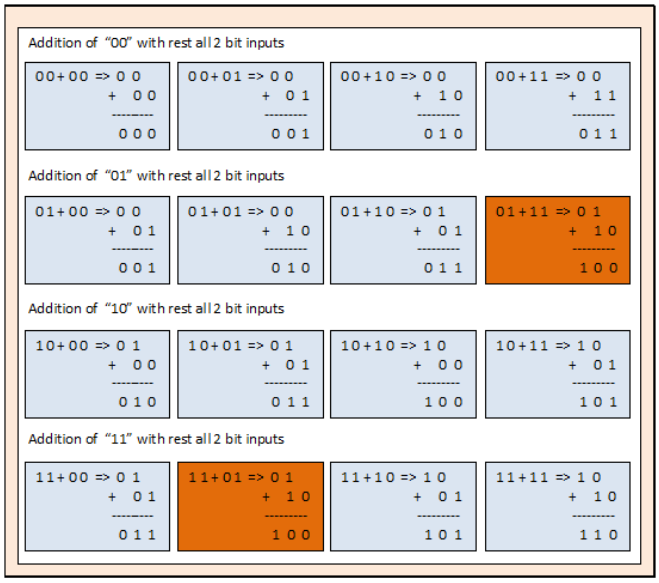

Fig. 3. Illustration of left-to-right addition for the binary system.

The algorithm used to develop the architecture for the left-to-right addition is:

Step1: Read two n-bit numbers A and B represented as:

$$A = A_{n-1}A_{n-2} \dots A_3A_2A_1A_0$$
$$B = B_{n-1}B_{n-2} \dots B_3B_2B_1B_0$$

Step2: Add the corresponding bits A_i and B_i to obtain the sum S_i and the curry C_i.

Step3: Add S_i to C_{i-1}, which performs MSB to LSB addition to produce the intermediate sum IS_m and the carry IC_m.

Step4: Add the previous carry C_{m-1} to the current sum S_m to produce the final output.

Consider the 4-bit binary numbers $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$, which are added to produce a 4-bit sum $S = S_3S_2S_1S_0$ with a carry out C_{out}. To start with the addition process, add the corresponding bits A_3 to B_3, A_2 to B_2, A_1 to B_1, and A_0 to B_0 simultaneously to produce the corresponding sums and carries. The addition of the sums and carries should be done from MSB to LSB in the left-to-right addition. Thus, S_3 is added to C_2, S_2 is added to C_1, S_1 is added to C_0. The sum bit S_0 and the last carry out C_3 are forwarded as they are. The addition result is correct if and only if no carry is generated. If the carry is generated, as portrayed in Figure 3, then the result must be corrected.

III. PROPOSED ARCHITECTURE

The proposed architecture of left-to-right addition using 8-bit addition is displayed in Figure 4. The addition process requires the addition of only two bits at a time, so only a Half Adder (HA) is used to design the architecture. The addition is carried out in two stages. The first stage, called the intermediate stage, generates the sum and the intermediate carry. The second stage, called the correction stage, is the correction of the sum generated in the first stage. If the two inputs of the 8-bit addition are $A = A_7A_6A_5A_4A_3A_2A_1A_0$ and $B = B_7B_6B_5B_4B_3B_2B_1B_0$, the first step is to add the corresponding bits A_i and B_i. Thus, A_7 is added to B_7, A_6 is added to B_6, and so on. Since only two bits A_i and B_i are added, a HA is used to produce the corresponding sum S_i and carry C_i. This addition is shown in the first row of Figure 4.

The next step in the left-to-right addition is to add the numbers from the most significant digit, where the MSB sum is added with the carry of the 'MSB-1' bit to produce the addition result. Thus, S_7 is added to C_6, S_6 is added to C_5 and the process continues to add the final sum bit S_1 to C_0. Furthermore, the last carry C_7 and the first sum bit S_0 are passed without change to the next stage. This addition process is exhibited in the second row of Figure 4. Again, since only two bits are being added, an HA is used to calculate the addition result, producing the sum bits IS_i and carry bits IC_i. If the carry is not generated as in the case shown in Figure 1, then the final result is a concatenation of C_7, the intermediate sum IS_7 to IS_1, and S_0. The above steps of the intermediate stage are highlighted in white in Figure 4. The carry may be generated in the left-to-right addition, as illustrated in Figure 3, and must be taken into account. This correction of the original sum with the carry is done in the correction stage of the architecture. The correction stage again adds the MSB sum with the carry of the 'MSB-1' bit to produce the addition result. Thus, IS_7 is added to IC_6, IS_6 is added to IC_5 and the process continues to add the last sum bit IS_3 to IC_2. The MSB carries from the intermediate stage are added together to obtain the final carry. The XOR gate is used in the second row of the correction stage as only the sum bit is needed.

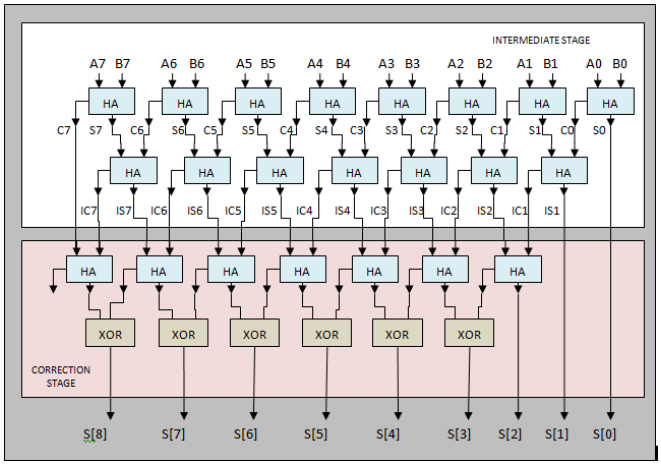

Fig. 4. Proposed architecture of 8-bit left-to-right addition.

An example of the proposed architecture is shown in Figure 5. Consider the addition of the numbers A = 10110111 and B = 11011101, which produce the result 110010100. When these two numbers are added, a carry is generated for each bit and therefore, when added using the available techniques, it takes more time to compute the result as the carry generated from the LSB has to be propagated to the MSB. As illustrated in Figure 5, in the addition process of the proposed architecture, a carry is generated for only 3 additions.

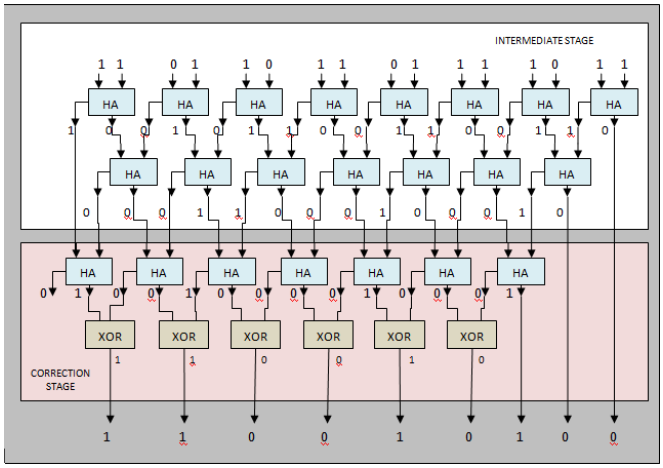

Fig. 5. Example of the proposed architecture for adding 10110111 and 11011101.

This novel architecture is inspired by left-to-right mental mathematics, specifically Vedic mathematics. To the best of our knowledge, this is the first architecture to adopt a left-to-right addition approach, which is traditionally used for rapid mental calculations. In contrast, all previously developed adder architectures follow the conventional method of starting the addition from the LSB and propagating the carry to higher order bits. Previous adder designs have aimed to optimize the addition architecture by either predicting the generated carry or pre-computing it using propagate, kill, and generate signals derived from the inputs. As a result, these architectures typically involve either increased hardware complexity or complicated carry propagation mechanisms. In contrast, this work introduces an entirely different approach to addition by processing the bits starting from the MSB to minimize carry generation. This simplifies the implementation of the addition operation. The proposed adder architecture relies solely on HA and XOR gates to produce the result without waiting for carry propagation, using simple logic. The design is both regular, symmetric and systematic, making it well suited for VLSI chip implementations. In addition, the architecture's depth remains constant regardless of the number of bits, allowing it to be seamlessly applied to n-bit addition.

IV. RESULTS AND DISCUSSION

The results of the proposed left-to-right adder are evaluated in three levels. The first level is the verification of the adder by simulation. The second level is the verification of the proposed adder architecture with the existing adder techniques in terms of time, power and area. The third level is the comparison of our work with the existing ones. The adder is designed for adding various binary bits using Verilog and the result of the addition has been verified for a wide range of data. The left-to-right adder has been simulated using the NC-Sim simulator for 8, 16, and 32 bits, and the simulation results of the 32-bit adder are shown in Figure 6. Second, the adder is designed using various techniques and the hardware cost is analyzed. The architecture is synthesized using the Cadence RTL compiler with 180 nm technology. The synthesis snapshots for 8-bit and 32-bit adders are shown in Figures 7 and 8, respectively.

The area, power, and time required to perform the addition operation are compared in Tables I, II, and III, respectively. The results show that the area occupied by the proposed Vedic adder is less than the area occupied by the parallel prefix adder, however, CSA occupies the least area among the adders as the number of addition bits increases. When comparing power and time, the proposed left-to-right adder outperforms the other adders. The proposed Vedic adder has constant time as the depth of the addition is fixed for all the bits. Also, it computes the addition in less time compared to the Kogge-Stone parallel prefix fast adder. Thus, the proposed architecture for the left-to-right adder performs best when power and time are the requirements for any embedded application. The area-delay product and the power-delay product of the proposed adder are presented in Table IV and Table V, respectively.

The comparison results disclose that the proposed Vedic adder has better power-delay and area-delay products compared to the existing techniques. As seen in Table V, the proposed left-to-right adder saves 8% power and about 23% area compared to the fast parallel prefix Kogge-Stone adder. For further analysis, the proposed left-to-right adder is compared with existing designs in Table VI. Although it occupies more area than the existing techniques, the proposed design is more suitable for applications that require low power addition operations in a shorter time.

TABLE I. AREA COMPARISON OF VARIOUS ADDERS

Adder	Area (um^2)		
	8-bit	16-bit	32-bit
CLA	1037.837	5575.762	13630.93
CSA	2328.084	2936.589	3339.706
Parallel prefix	997.92	2754.259	4929.345
Proposed adder	951.35	2042.41	4224.528

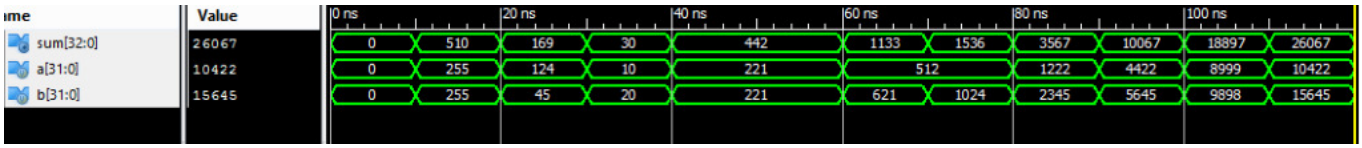

Fig. 6. Simulation results for 32-bit addition.

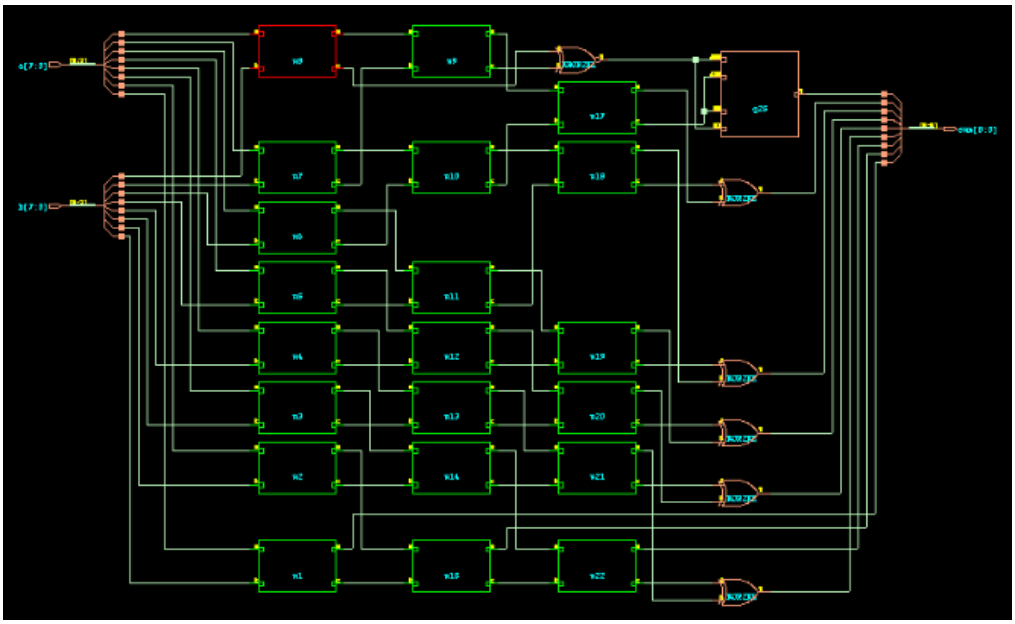

Fig. 7. Synthesis results for 8-bit addition.

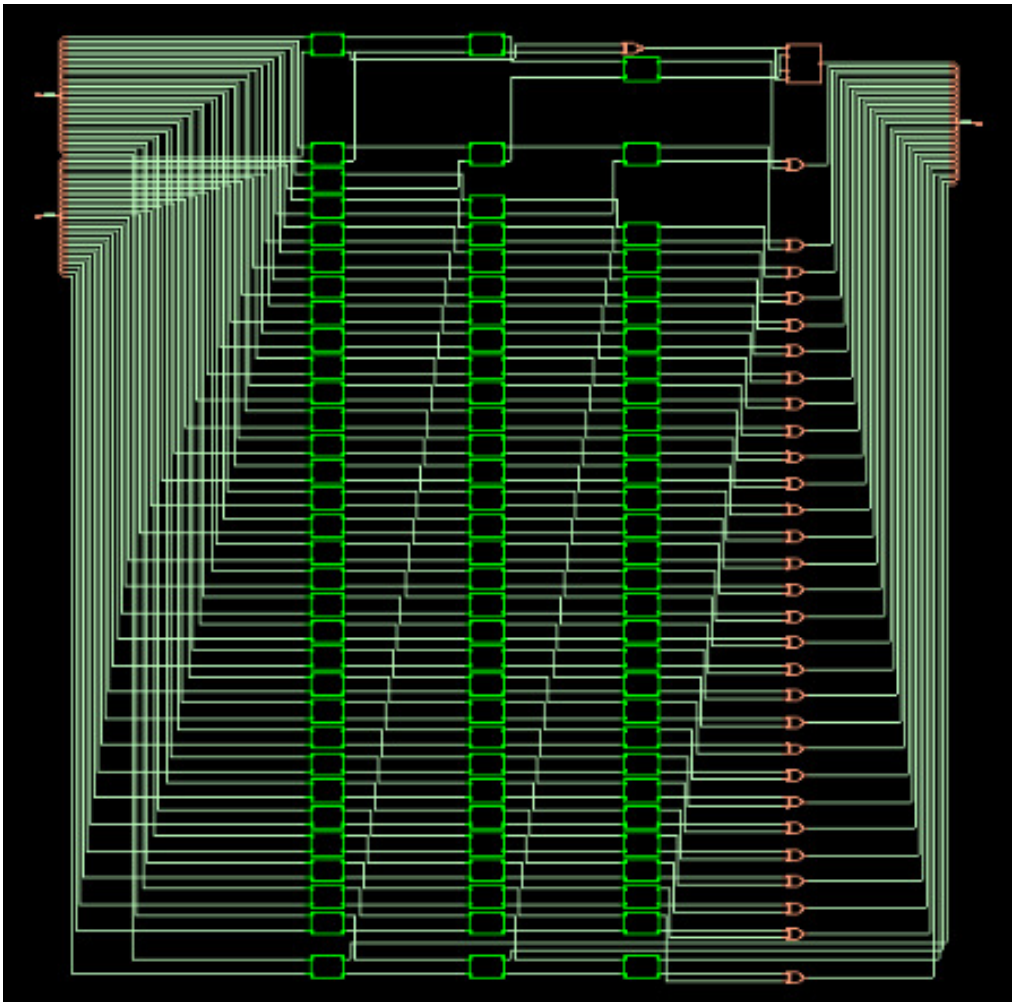

Fig. 8. Synthesis results for 32-bit addition.

TABLE II. POWER COMPARISON OF VARIOUS ADDERS

Adder	Power (pW)		
	8-bit	16-bit	32-bit
CLA	66496.21	161690	323379.1
CSA	54576.94	118646.3	249163
Parallel prefix	48830.78	162586.9	276343.1
Proposed adder	50461.43	113757.8	238693.5

TABLE III. TIME COMPARISON OF VARIOUS ADDERS

Adder	Time (ps)		
	8-bit	16-bit	32-bit
CLA	2115	2740	4015
CSA	1735	5511	11087
Parallel prefix	1857	2734	3611
Proposed adder	1150	1150	1150

TABLE IV. PERFORMANCE PARAMETERS OF VARIOUS ADDERS

Adder	Power × Delay	Area × Delay
CLA	0.0129 pJ	54.72 p
CSA	0.00276 pJ	37.02 p
Parallel prefix	0.99 fJ	17.79 p
Proposed adder	0.27 fJ	p

TABLE V. PERFORMANCE PARAMETERS OF VARIOUS ADDERS NORMALIZED TO CLA

Adder	Power × Delay	Saving (%)	Area × Delay	Saving (%)
CLA	1		1	
CSA	0.21	79	0.68	32
Parallel prefix	0.076	92.4	0.32	68
Proposed adder	0.02	98	0.089	91.91

The comparison results show that the proposed Vedic adder has better power-delay and area-delay products compared to the existing techniques. As evidenced in Table V, the proposed left-to-right adder saves 8% power and about 23% area compared to the fast parallel prefix Kogge-Stone adder. For further analysis, the proposed left-to-right adder is compared with existing designs in Table VI. Although it occupies more area than the existing techniques, the proposed design is more suitable for applications that require low power addition operations in a shorter time.

TABLE VI. COMPARISON OF THE PROPOSED 32-BIT ADDER WITH EXISTING 32-BIT ADDERS

Adder	Technology used	Area (um^2)	Power (uW)	Delay
Modulo (2n+1) adder [10]	90 nm	1308	382	464 ns
Diminished-1 modulo (2n+1) adder [11]	60 nm	944	272	-
Jackson valency–4 [9]	180 nm	1091	25.302	1546 ps
Jackson valency–5 [9]	180 nm	1112	25.844	1563 ps
Proposed adder	180 nm	4224.5	0.2387	1150 ps

V. CONCLUSION

The proposed left-to-right adder has been thoroughly evaluated at three levels: functional verification, hardware performance analysis, and comparative benchmarking against existing adder architectures. The results demonstrate that the proposed design offers significant advantages in specific metrics, making it highly suitable for embedded and power-sensitive applications. The functionality of the adder was verified by simulation for 8-bit, 16-bit, and 32-bit addition using NC-Sim. The simulation results confirmed accurate addition across a wide range of input data. Notably, the left-to-right addition approach effectively reduced carry propagation, simplifying the addition process. The hardware performance of the proposed adder was also analyzed. The adder architecture was synthesized using Cadence RTL compiler with 180 nm technology. The synthesis results exhibit that the proposed adder achieves better power efficiency and time performance compared to conventional adders. The depth of the architecture remains constant regardless of the bit size, allowing for scalable and predictable performance.

Tables I, II, and III highlight the key performance metrics, area, power, and time, for the proposed adder compared to Carry-Lookahead Adders (CLA), Carry-Save Adders (CSA), and parallel prefix adders. The results reveal that the proposed adder occupies slightly more area than CSA, but significantly less area than parallel prefix adders for all bit sizes. The proposed adder also reduces the power consumption by 8% compared to the parallel prefix adder. With a constant delay of 1150 ps for 8-bit, 16-bit, and 32-bit additions, the proposed adder outperforms other architectures, achieving faster computation times. Tables IV and V summarize the power-delay and area-delay product metrics, highlighting the superiority of the proposed design. For a 32-bit addition:

- Power-delay product: The proposed adder achieves a 98% saving compared to CLA and a 92.4% saving compared to parallel prefix adders.
- Area-delay product: The proposed adder achieves a 91.91% saving compared to CLA and a 68% saving compared to parallel prefix adders.

As shown in Table VI, the proposed 32-bit adder demonstrates competitive performance when compared to other advanced designs, such as Modulo (2n+1) and Jackson valency adders. While its area usage is higher due to the nature of the architecture, it excels in power efficiency and achieves significantly lower delay times, making it ideal for low-power, high-speed applications. The proposed left-to-right adder introduces an innovative approach inspired by Vedic mathematics, leveraging its constant depth and reduced carry propagation to achieve optimized performance. Its exceptional power-delay and area-delay characteristics make it an excellent choice for embedded systems and other computationally intensive applications where efficiency is paramount. Future work could explore further refinements in area optimization and integration with larger-scale systems.

The primary limitation of this architecture is that the adder produces incorrect results for certain data combinations due to its fixed depth. Specifically, in a 4-bit adder, errors occur when the carry propagates beyond three bits, resulting in an output of exactly 16 in three cases: adding 1 with 15, 3 with 13, and 5 with 11. Among the 135 possible unique additions, three exhibit errors, yielding an accuracy of 97.7%.

A similar issue arises in 8-bit, 16-bit, and 32-bit adders when the carry propagation exceeds three bits. To accurately add the results, the correction stage depth can be increased to (N-1) stages, where N represents the number of bits being added. Since the correction stage depth is increased, the area increase is about 30%. Hence future research could focus on enhancing area and power optimization by integrating Vedic addition principles with multiplexer based design approach to accurately compute the addition.

ACKNOWLEDGEMENT

We would like to acknowledge Prof. H. T. Vergos for his thorough remarks and for pointing out the limitations of the proposed design.

REFERENCES

[1] V. Vijay *et al.*, "A Review On N-Bit Ripple-Carry Adder, Carry-Select Adder And Carry-Skip Adder," *Journal of VLSI Circuits and Systems*, vol. 4, no. 1, pp. 27–32, Jun. 2022, https://doi.org/10.31838/jvcs/04.01.05.

[2] V. Pudi and K. Sridharan, "Low Complexity Design of Ripple Carry and Brent–Kung Adders in QCA," *IEEE Transactions on Nanotechnology*, vol. 11, no. 1, pp. 105–119, Jan. 2012, https://doi.org/10.1109/TNANO.2011.2158006.

[3] A. S. Hameed and M. J. Kathem, "High speed modified carry save adder using a structure of multiplexers," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1591–1598, Apr. 2021, https://doi.org/10.11591/ijece.v11i2.pp1591-1598.

[4] M. S. Hossain and F. Arifin, "Design and Evaluation of a 32-bit Carry Select Adder using 4-bit Hybrid CLA Adder," *AIUB Journal of Science and Engineering (AJSE)*, vol. 20, no. 2, pp. 1–7, May 2021, https://doi.org/10.53799/ajse.v20i2.119.

[5] K. Chirca *et al.*, "A static low-power, high-performance 32-bit carry skip adder," in *Euromicro Symposium on Digital System Design, 2004. DSD 2004*, Rennes, France, 2004, pp. 615–619, https://doi.org/10.1109/DSD.2004.1333335.

[6] F. K. Gurkayna, Y. Leblebicit, L. Chaouati, and P. J. McGuinness, "Higher radix Kogge-Stone parallel prefix adder architectures," in *2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, 2000, pp. 609–612 vol.5, https://doi.org/10.1109/ISCAS.2000.857516.

[7] J. Lee, H. Seo, H. Seok, and Y. Kim, "A Novel Approximate Adder Design Using Error Reduced Carry Prediction and Constant Truncation," *IEEE Access*, vol. 9, pp. 119939–119953, 2021, https://doi.org/10.1109/ACCESS.2021.3108443.

[8] D. S. Manikanta, K. S. S. Ramakrishna, M. Giridhar, N. Avinash, T. Srujan, and R. S. R, "Hardware Realization of Low power and Area Efficient Vedic MAC in DSP Filters," in *2021 5th International Conference on Trends in Electronics and Informatics*, Tirunelveli, India, 2021, pp. 46–50, https://doi.org/10.1109/ICOEI51242.2021.9453041.

[9] N. Poornima and V. S. K. Bhaaskaran, "Design and Implementation of 32-Bit High Valency Jackson Adders," *Journal of Circuits, Systems and Computers*, vol. 26, no. 07, Jul. 2017, Art. no. 1750123, https://doi.org/10.1142/S0218126617501237.

[10] H. T. Vergos and G. Dimitrakopoulos, "On Modulo 2^n+1 Adder Design," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 173–186, Feb. 2012, https://doi.org/10.1109/TC.2010.261.

[11] S. K. Singhal, B. K. Mohanty, S. K. Patel, and G. Saxena, "Efficient Diminished-1 Modulo (2n+1) Adder Using Parallel Prefix Adder," *Journal of Circuits, Systems and Computers*, vol. 29, no. 12, Sep. 2020, Art. no. 2050186, https://doi.org/10.1142/S0218126620501868.

[12] J. M R and Manjudevi, "Analysis of Special Adders for Digital System," in *2021 5th International Conference on Trends in Electronics and Informatics*, Tirunelveli, India, 2021, pp. 253–256, https://doi.org/10.1109/ICOEI51242.2021.9452953.

[13] A. K. Panda, R. Palisetty, and K. C. Ray, "High-Speed Area-Efficient VLSI Architecture of Three-Operand Binary Adder," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3944–3953, Nov. 2020, https://doi.org/10.1109/TCSI.2020.3016275.

[14] P. Jayakrishna, P. Sravani, R. S. Reddy, and S. Ashritha, "Design of High-Speed Area-Efficient VLSI Architecture of 32-Bit Three-Operand Binary Adder," *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. VI, pp. 1353–1361, Jun. 2023, https://doi.org/10.22214/ijraset.2023.53550.

[15] K. Deepthi, P. Bhaskar, M. Priyanka, B. V. Sonika, and B. N. Shashikala, "Design and Implementation of High-Speed Low-Power Carry Select Adder," in *Cognitive Informatics and Soft Computing: Proceeding of CISC 2020*, Balasore, Odisha, India, 2020, pp. 517–530, https://doi.org/10.1007/978-981-16-1056-1_41.

[16] Y.-Y. Chu, S.-H. Shieh, H. Feng, H. Deng, M.-S. Shiau, and D.-C. Huang, "A High-Speed Carry-Select Adder with Optimized Block Sizes," in *2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification*, Xiamen, China, 2021, pp. 182–186, https://doi.org/10.1109/ASID52932.2021.9651488.

[17] P. Kavipriya, S. Lakshmi, T. Vino, M. R. Ebenezar Jebarani, and G. Jegan, "Booth Multiplier Design Using Modified Square Root Carry-Select-Adder," in *2021 International Conference on Artificial Intelligence and Smart Systems*, Coimbatore, India, 2021, pp. 1647–1653, https://doi.org/10.1109/ICAIS50930.2021.9396032.

[18] O. Anjaneyulu and C. V. K. Reddy, "A novel design of full adder cell for VLSI applications," *International Journal of Electronics*, vol. 110, no. 4, pp. 670–685, Apr. 2023, https://doi.org/10.1080/00207217.2022.2059819.

[19] V. Vijay, J. Prathiba, S. N. Reddy, and V. R. Rao, "Energy efficient CMOS Full-Adder Designed with TSMC 0.18 µm Technology," in *International Conference on Technology and Management*, Hyderabad, India, 2011, pp. 356–361.

AUTHORS PROFILE

M. G. Anuradha received a B.E degree from Visvesvaraya Technological University in Electronics and Communication in 2002, an M.Tech degree in VLSI and Embedded System Design from Visvesvaraya Technological University in 2008, and a Ph.D degree from Visvesvaraya Technological University in 2022. She is currently working as an associate professor in the Department of Electronics and Communication in JSS Academy of Technical Education, Bengaluru. She has published more than 20 research articles in various international conferences and journals. Her research interests are in the areas of circuit analysis, digital and analog circuit design, pattern recognition, image clustering and its related VLSI architectures.

R. Arun Kumar received a B.E degree from Visvesvaraya Technological University in Electronics and Communication and an M.Tech degree in VLSI and Embedded System Design from Visvesvaraya Technological University in 2011. He is currently working as an admin head and math teacher in Ohana School, Bengaluru. His research interests are circuit analysis, various mathematical algorithms, and their related VLSI architectures.