iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84201

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Design and Implementation of a High-Speed Three-Operand Binary Adder Using Parallel Prefix Architectures

S Aparna[1], C. Padma[2]
[1]*M.Tech Scholar,* [2]*Associate Professor, Dept. of ECE Sri Venkateswara College of Engineering, Tirupati, Andhra Pradesh, India*

Abstract: ***High-speed and space-efficient arithmetic units are necessary for the hardware implementation of cryptographic algorithms to guarantee security and performance. Efficient multi-operand addition, especially three-operand binary addition, is crucial for modular operations like multiplication and exponentiation. A high-speed, low-area three-operand binary adder for cryptography and pseudorandom bit generator (PRBG) applications is presented in this study. Kogge-Stone, Han-Carlson, and Ladner-Fischer are examples of parallel prefix adders that are used to increase throughput and decrease propagation latency. For VLSI systems, heterogeneous delay-insensitive coding is used to further optimize power, area, and performance. A Carry Look-Ahead (CLA) adder, which lowers critical route delay through parallel carry generation, is added to the design to improve carry calculation. When compared to traditional designs, the suggested hybrid architecture provides increased speed and efficiency. Its usefulness for high-performance computing and digital signal processing applications is demonstrated via implementation using Xilinx Vivado.***

Keywords—Three-Operand Binary Adder, Modular Arithmetic, Cryptographic Hardware, Carry Look-Ahead Adder (CLA), Parallel Prefix Adders, VLSI Design.

I. INTRODUCTION

Multi-operand adders are important for arithmetic design blocks, especially in the summation of partial products of hardware multipliers. Multi-operand adders are required for building multipliers where multiple partial products must be added up to have the multiplication result. Since most of the power is consumed at the arithmetic blocks, especially at multipliers, lower-power devices with lower switching noise are desirable [2].

Among all adders, RCA has less area and energy with higher delay, whereas other adders (fast adders) have less delay with higher area and energy. On the other side, area and energy are the two major constraints in most of the systems that demand the addition of a large number of operands. The Binary tree adder employed in these systems should be area- and energy-efficient.

The operands considered for addition can be single bit or of multiple bits; thus, the input and output of the adder can be in multiple bits [3]. A multi-operand adder can be represented simply by an architecture comprising a compressor tree, which reduces the partial sum and propagated carry [3]. There are numerous types of MTA, but the adders used in this work are the Array tree adder, Wallace tree adder, Balanced delay tree adder and Overturned-stairs tree adder. For any arithmetic functioning unit, adders are the most important components because of their resource consumption and the delay involved in processing. Adders also form a part of multipliers, which are another resource-intensive component of arithmetic circuits [3].

The simplest MOA is a binary tree adder (BTA), designed by connecting two-operand adders in a binary tree configuration for the addition of multiple operands [5]. The delay and energy efficiency of the Binary tree adder structure depend on the performance of adders used in the structure. There are numerous types of adders such as ripple carry adder (RCA), carry-look-ahead adder (CLA), parallel prefix adder (PPA), carry-select adder (CSLA) and carry-skip adder (CSKA), which can be used to develop the BTA structure, where each adder has its own trade-offs between area, delay and energy consumption.

As a result, binary addition is so important that any improvement in it can result in a performance gain for a computer system Among all adders, RCA has less area and energy with higher delay, whereas other adders (fast adders) have less delay with higher area and energy. On the other side, area and energy are the two major constraints in most of the systems that demand the addition of a large number of operands. The Binary tree adder employed in these systems should be area- and energy-efficient., and so help to enhance the whole system's performance. The carry chain is the most significant issue in binary addition. The length of the carry chain grows in proportion to the breadth of the input operand.

The worst-case scenario is when the carry takes the longest possible path from the least significant bit (LSB) to the most significant bit (MSB). It is possible to accelerate, but not eliminate, the carry chain to increase the performance of carry propagate adders. As a result, when it comes to optimising computer architecture, most digital designers turn to faster adders because they tend to determine the critical route for most computations. In most digital circuit designs, such as digital signal processors (DSP) and microprocessor data route units, the binary adder is a key component. As a result, substantial research is still being done to improve the adder's power-delay performance. Three-operand adders are known to offer the best performance in VLSI implementations. Reconfigurable logic, such as Tanner EDA, has grown in popularity in recent years as a result of its superior performance in terms of speed and power over DSP-based and microprocessor-based solutions for a variety of practical designs involving mobile DSP and telecommunications applications. With the growing popularity of mobile and portable devices, which make full use of DSP functions, the power advantage is especially essential. Three-operand adders, on the other hand, will operate differently than VLSI versions due to the topology of the programmable logic and routing resources on FPGAs. Most contemporary FPGAs, in particular, use a fast-carry chain that streamlines the carry path for the simple Ripple Carry Adder(RCA).

II. LITERATURE SURVEY

Digital systems like multipliers and filters, particularly in medical IoT devices, frequently use multi-operand addition. When conventional adders are used to sum intermediate results, the area and delay are frequently increased. This is addressed by using a compressor tree and multi-operand addition approaches to minimize area overhead while reducing delay [1]. To further enhance performance in terms of area, latency, and power consumption, multiplication techniques like the Wallace tree (WT) multiplier and the add-and-shift approach are used [6].

Different trade-offs are seen in adder architectures. Low latency is provided by Carry Look-Ahead Adders (CLA) and Parallel Prefix Adders (PPA), but they need a lot of space and energy [7–14]. Ripple Carry Adders (RCA), on the other hand, have higher delays but use less space and power [15,16]. The Carry Select Adder (CSLA), which offers increased speed with moderate area and power consumption, is designed to balance these concerns [17]. Additional CSLA optimizations, like Binary to Excess-1 Converter (BEC)-based designs and Square Root CSLA, aid in lowering area and power usage [18–20].

Chris Wallace created the Wallace Tree Adder (WTA), an effective multi-operand addition mechanism. Carry Save Adders (CSA) are used to reduce several operands to two before a final addition step [15]. CSAs are very efficient since they only add one full-adder delay. Binary Tree Adders (BTA) have a larger delay than Wallace tree structures, while being commonly utilized because of their straightforward and uniform construction.

Tanner EDA tools are used to implement and evaluate three-operand adders and tree-based adders [5]. To assess performance, these implementations are contrasted with Han-Carlson Adders (HCA) and Carry Save Adders (CSA). The paper identifies real-world difficulties in creating tree-based adders and makes recommendations for ways to increase system performance, speed, and area efficiency.

Logical irreversibility in traditional computing systems causes physical irreversibility, which produces heat. Each irreversible process will inevitably dissipate energy on the scale of kT each cycle. Although it restricts efficiency, this energy dissipation guarantees signal stability. Logically reversible computation, which allows computations to be reversed without information loss and greatly reduces energy dissipation, is suggested as a solution to this problem.

Information loss is prevented by reversible logic, which guarantees a one-to-one mapping between inputs and outputs. The computation process includes output generation, backward calculation to eliminate undesired data, and forward computation with storage of intermediate results. Reversible logic gates are becoming more and more common in low-power VLSI design, quantum computing, nanotechnology, and optical systems. Tools like Xilinx and Verilog HDL are used to assess their performance in terms of power, delay, and area.

Floating-point multipliers and reversible ALUs are examples of recent innovations. Methods like reversible Wallace tree multipliers and operand decomposition maximize garbage outputs, quantum cost, and delay. All things considered, combining reversible logic with multi-operand addition methods offers a practical way to create high-performance, low-power systems for uses like advanced computing architectures and medical IoT.

III. PREVIOUS WORK

The main goals of current multi-operand addition research are to increase arithmetic circuit speed, size, and power efficiency. Due to their straightforward design and minimal area consumption, conventional methods like Ripple Carry Adders (RCA) are frequently employed; nevertheless, because of the sequential carry chain, they have a considerable propagation delay.

Faster adders like Carry Look-Ahead Adders (CLA) and Carry Select Adders (CSLA), which lower latency at the expense of more hardware complexity and power consumption, have been developed to get around this restriction.

Three-operand addition frequently uses Carry Save Adders (CSA), especially in multiplier designs and cryptography applications. By producing partial sum and carry outputs concurrently, CSA minimizes intermediate carry propagation delay. Nevertheless, carry propagation via a traditional adder is still necessary for the last stage, which restricts overall speed and adds delay. Wallace Tree Adders (WTA) use carry-save techniques in a tree structure to further increase speed, but because of their irregular layout, they are less adaptable and more difficult to construct for different operand sizes.

The logarithmic delay characteristics of Parallel Prefix Adders (PPA) like Kogge-Stone, Han-Carlson, and Ladner-Fischer architectures have drawn interest. These adders compute carry signals in parallel, greatly reducing the critical path delay. Kogge-Stone has the quickest performance, but because of its complicated nodes and heavy wiring, it takes up a lot of space. For realistic VLSI implementations, Han-Carlson and Ladner-Fischer provide a superior trade-off between speed and hardware resources.

Even with these improvements, current designs still struggle to balance power, area, and delay at the same time. While area-efficient designs sacrifice speed, many high-speed adders increase performance at the expense of more complex hardware. An efficient three-operand adder architecture that delivers high speed with low area and power overhead is therefore required, especially for applications in high-performance computing and cryptography.

IV. EXISTING WORK

It is found that there is no notable attempt made in the literature for the delay refinement of RCA-BTA. Therefore, to explore the possibilities for delay minimisation of the RCA-BTA structure, the critical path delay analysis is presented [5]. It is observed that most of the work proposed in the literature is mainly focused on improving the design metrics of fast adders, which can enhance the performance of the WTA structure. From [5], it is clear that the WTA has less delay in comparison to RCA-BTA, but its structural irregularity requires the redesign of the WTA for different values of N. On the other hand, the BTA structure is simple and regular, which can easily be designed for any value of N. Although the delay of RCA-BTA can be improved by minimizing RCA delay, the WTA performance can be enhanced by the usage of an efficient fast adder at the final addition stage. Different types of adders are as follows:

A. Half Adder and Full Adder

A half adder performs the addition of two 1-bit inputs, producing a sum $S = A \oplus B$ and carry $C = AB$. A full adder extends this functionality by including a carry-in input. C_{in}, generating outputs:

$S = A \oplus B \oplus C_{in}, C_{out} = AB + (A \oplus B)C_{in}$.

A full adder can be implemented using two half adders and an OR gate.

B. Ripple Carry Adder (RCA)

An N-bit RCA is formed by cascading full adders, where the carry propagates sequentially. Although area-efficient ($O(n)$), it suffers from linear delay due to carry propagation:

$S_i = A_i \oplus B_i \oplus C_i, C_{i+1} = A_iB_i + (A_i + B_i)C_i$.

C. Carry Skip Adder (CSKA)

The CSKA improves RCA performance by allowing carry to bypass blocks of bits when propagation conditions are satisfied. The delay depends on block size and is reduced compared to RCA. Variable block designs further optimize latency at the cost of increased hardware.

D. Carry Select Adder (CSA)

The carry select adder computes sum and carry for both possible carry-in values (0 and 1) in parallel and selects the correct output using multiplexers. This reduces delay but increases area due to duplicated hardware.

E. Carry Look-Ahead Adder (CLA)

The CLA reduces carry propagation delay by precomputing generate and propagate signals: This enables parallel carry computation, achieving significantly lower delay compared to RCA.

$G_i = A_iB_i, P_i = A_i \oplus B_i.$

F. Carry Save Adder (CSA)

A carry-save adder is used for multi-operand addition. It produces partial sum and carry vectors without immediate carry propagation, thereby improving speed. Final addition is performed using a conventional adder.

G. Parallel Prefix Adders

Advanced adders such as Kogge-Stone, Brent-Kung, and Han-Carlson further reduce delay to $O(\log n)$ using parallel prefix structures. These architectures offer high speed at the expense of increased area and complexity. Trade-offs between latency, area, and power are offered by various adder architectures. CLA and parallel prefix adders provide high speed, while *RCA* is sluggish but area-efficient. Carry-save and carry-select adders are useful for enhancing performance in high-speed and multi-operand applications, respectively. The design restrictions unique to a given application determine which adder is best.

Three-Operand Adder

In this study, a unique VLSI design utilizing a parallel prefix-based adder for three-operand addition in modular arithmetic is presented. The suggested architecture uses a four-stage structure to effectively calculate the total of three n-bit operands, in contrast to traditional prefix adders. Bit-Addition Logic, Base Logic, Propagate-Generate (PG) Logic, and Sum Logic are the steps.

- Bit-Addition Logic: Three input operands, A, B, and C, are bitwise added in the first step by an array of complete adders. Every complete adder produces intermediate sum (Si) and carry (Ci) signals. This step reduces the initial computation delay by enabling concurrent processing of all input bits.
- Base Logic: The second stage uses the intermediate sum and carry outputs to calculate the propagate (Pi) and generate (Gi) signals. Each base cell generates Pi and Gi by taking the previous carry and the current total. The first level of this logic incorporates an external carry-in (Cin), which yields n + 1 n+1 base cells. Signals are prepared for effective carry computation at this stage.
- PG Logic (Prefix Computation): The third step computes carries effectively by using concurrent prefix structures. Compared to traditional ripple-based architectures, carry propagation delay is greatly decreased by utilizing prefix operations. Logarithmic delay performance is guaranteed by the prefix network.
- Sum Logic: The final sum outputs are produced by combining the computed carry signals with propagate signals in the last stage. The three-operand addition procedure is finished at this point.

In digital systems, addition is a crucial function that directly affects total performance. Sequential carry propagation causes a large propagation latency for conventional Ripple Carry Adders (RCA). Carry Look-Ahead Adders (CLA) enhance area at the cost of speed. By lowering delay to O(log n), Parallel Prefix Adders (PPA) provide an ideal trade-off.

By precalculating outputs for multiple carry inputs, Carry Select Adders (CSLA) further increase speed but necessitate more hardware. To maximize latency, area, and power, recent research focuses on hybrid designs that combine prefix logic with carry-skip algorithms.

V. METHODOLOGY AND IMPLEMENTATION

By implementing a hybrid variable latency method, the Variable Stage Structure (VSS) aims to balance the critical path delay. This is accomplished by substituting a Parallel Prefix Adder (PPA)-based structure for the nucleus stage. The suggested hybrid architecture is divided into several stages (1 to Q). The first stage uses half adders to implement a Ripple Carry Adder (RCA), and the stages that follow incorporate concatenation and incrementation blocks in addition to RCA and skip logic. The middle stage is swapped out for a PPA to increase speed because it adds the most latency and area. Pseudo-generate and propagate functions are used to carry out carry generation in the proposed PPA, which combines Ling adder and Kogge-Stone adder architectures. Compared to traditional adders, this improves calculation performance and lowers logic complexity. Preprocessing, prefix computation, and post-processing are the three phases of the parallel prefix network's implementation. Intermediate signals lower logic levels in the Ling-based prefix structure, and the Kogge-Stone adder guarantees quick carry computation, reducing sum and carry generation latency.

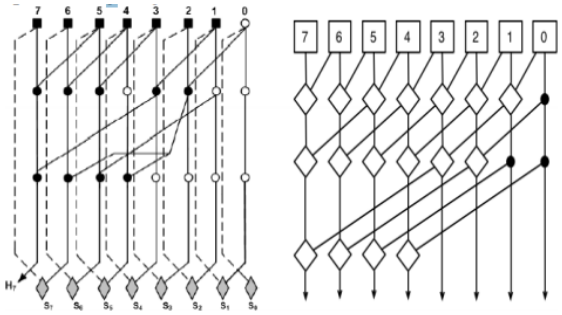


Fig. 1 Internal structure of Ling Adder and Kogge-Stone

Figure 1 depicts the internal structure of an 8-bit PPA based on Ling equations and Kogge-Stone Adder. The proposed structure is developed for 16 bits. In this method, there are three stages for Ling carry and sum computation. The nodes represented with black squares and circles are used for the computation of carry-generate and propagate functions. The diamond structure is responsible for the final stage sum computation. The difference between the conventional schemes and PPA lies in the second stage, which is responsible for carry generation. Ling adders use an intermediate representation, which becomes an advantage because the number of logic levels is reduced. Kogge-Stone is known to be the fastest adder among others in terms of carry computation.

Fig. 2 PPA Carry and Sum generation

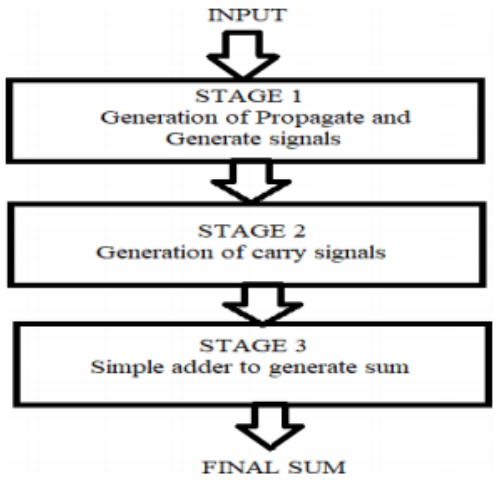


Fig. 2 PPA Carry and Sum generation

High-speed VLSI systems frequently employ the Carry Select Adder (CSLA) to minimize carry propagation time. It uses multiplexers to choose the final output after computing partial sums in parallel for carry inputs Cin = 0 and Cin = 1. However, because of redundant RCA structures, the traditional CSLA has significant power and area consumption. To solve this problem, a Binary to Excess-1 Converter (BEC) modified CSLA is suggested, in which a BEC unit is used in place of the RCA corresponding to Cin = 1. This increases area and power efficiency while drastically lowering the number of logic gates. As a conditional sum adder, the CSLA selects and precomputes results dependent on the carry input. Faster operation is made possible by the use of multiplexers, but this comes at the expense of more hardware, which the BEC technique helps to offset.

By sharing logic between carry-in circumstances, a Common Boolean Logic (CBL)-based SQRT CSLA is introduced to further optimize the system by removing superfluous adder cells. Reusing XOR and inverter logic is made possible by the complementary sum outputs for Cin = 0 and Cin = 1, according to an analysis of the complete adder truth table. Similarly, shared AND and OR gates are used to generate carry outputs. Compared to traditional and BEC-based CSLA devices, this method achieves a better power-delay product by lowering the number of transistors and increasing speed.

Concurrent error detection approaches are used to address reliability in addition to performance enhancement [1]–[7]. Conventional adders are restricted to single fault detection and use internal comparisons or parity prediction to identify mistakes. A concurrent error-detecting adder with improved testability is suggested as a solution to this problem.

The design is based on a multi-block CSLA architecture [20], in which two candidate outputs corresponding to distinct carry inputs are computed by each block. By producing double carry outputs and predicted parity (pS), the suggested adder makes error detection possible by comparing predicted and real parity values. This method increases dependability in critical systems and guarantees the discovery of errors brought on by single stuck-at problems.

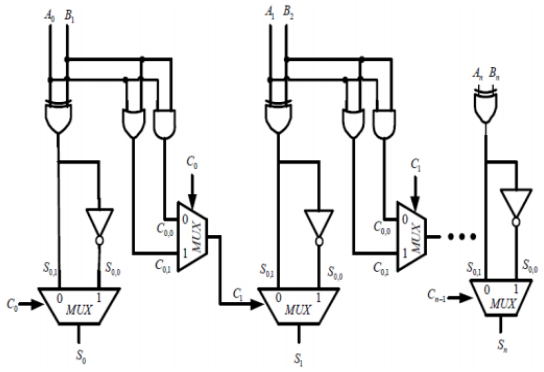


Fig. 3: Internal structure of the proposed area-efficient carry select adder is constructed by sharing the common Boolean logic term.

Additionally, the suggested design provides C-testability, which permits full fault coverage with a fixed number of test patterns independent of bit-width. In particular, just ten test patterns are needed to find errors throughout the adder structure. To guarantee that errors spread to observable outputs without masking, the design uses modified components like HA', INC', and MUX'. Results from fault simulations prove concurrent error detection capability and confirm 100% test coverage. The suggested architecture offers notable benefits in reliability, testability, and high-speed operation, making it appropriate for contemporary VLSI systems even though it adds about 70% hardware overhead for a 32-bit version.

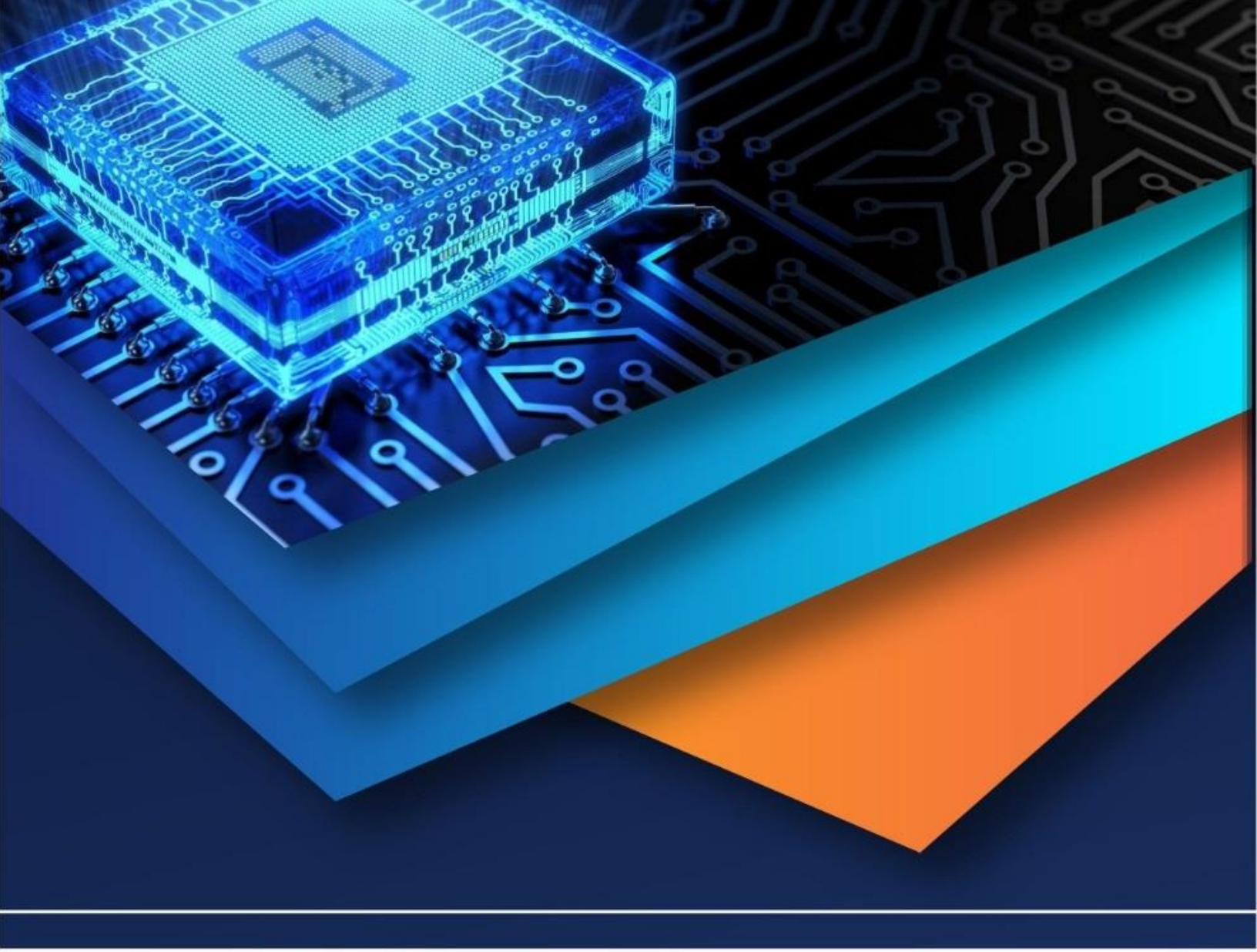
VI. CONCLUSION

In this paper, a high-speed, area-efficient adder technique and its VLSI architecture are proposed to perform the three-operand binary addition for efficient computation of modular arithmetic used in cryptography and PRBG applications. The proposed three-operand adder technique is a parallel prefix adder that uses four-stage structures to compute the addition of three input operands. The novelty of this proposed architecture is the reduction of delay and area in the prefix computation stages in PG logic and bit-addition logic, which leads to an overall reduction in area and delay. For a fair comparison, the concept of a hybrid Han-Carlson two-operand adder is extended to develop a hybrid Han-Carlson three-operand adder (HHC3A) topology. The same coding style adopted in the proposed adder architecture is extended to implement the hybrid Han-Carlson three-operand adder, Kogge Stone, using Verilog HDL. Further, all these designs are synthesized using the commercially available Spartan 3 technology library to obtain the core area and delay timing. Application-wise, we implemented our adder in the adder part of an FIR filter for further efficiency in the field of area and delay. Concluding that our adder was comparatively better than other adders by considering both the area and delay.

REFERENCES

[1] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, “FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field,” IEEE Access, vol. 7, pp. 178811–178826, 2019.
[2] Z. Liu, J. GroBschadl, Z. Hu, K. Jarvinen, H.Wang, and I. Verbauwhede, “Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the Internet of Things,” IEEE Trans. Comput., vol. 66, no. 5, pp. 773–785, May 2017.
[3] Z. Liu, D. Liu, and X. Zou, “An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor,” IEEE Trans. Ind. Electron., vol. 64, no. 3, pp. 2353–2362, Mar. 2017.
[4] B. Parhami, Computer Arithmetic: Algorithms and Hardware Design. New York, NY, USA: Oxford Univ. Press, 2000.
[5] P. L. Montgomery, “Modular multiplication without trial division,” Math. Comput., vol. 44, no. 170, pp. 519–521, Apr. 1985.
[6] S.-R. Kuang, K.-Y. Wu, and R.-Y. Lu, “Low-cost high-performance VLSI architecture for montgomery modular multiplication,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 2, pp. 434–443, Feb. 2016.
[7] S.-R. Kuang, J.-P. Wang, K.-C. Chang, and H.-W. Hsu, “Energy-efficient high-throughput montgomery modular multipliers for RSA cryptosystems,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 11, pp. 1999–2009, Nov. 2013.
[8] S. S. Erdem, T. Yanik, and A. Celebi, “A general digit-serial architecture for montgomery modular multiplication,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1658–1668, May 2017.

[9] R. S. Katti and S. K. Srinivasan, “Efficient hardware implementation of a new pseudo-random bit sequence generator,” in Proc. IEEE Int. Symp. Circuits Syst., Taipei, Taiwan, May 2009, pp. 1393–1396.
[10] A. K. Panda and K. C. Ray, “Modified dual-CLCG method and its VLSI architecture for pseudorandom bit generation,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 3, pp. 989–1002, Mar. 2019.
[11] A. Kumar Panda and K. Chandra Ray, “A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation,” IEEE Trans. Instrum. Meas., vol. 69, no. 4, pp. 1011–1019, Apr. 2020.
[12] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design—A Systems Perspective. Reading, MA, USA: Addison-Wesley, 1985.
[13] T. Kim, W. Jao, and S. Tjiang, “Circuit optimization using carry-save adder cells,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 17, no. 10, pp. 974–984, Oct. 1998.
[14] A. Rezai and P. Keshavarzi, “High-throughput modular multiplication and exponentiation algorithms using multibit-scan– multibit-shift technique,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 9, pp. 1710–1719, Sep. 2015.
[15] A. K. Panda and K. C. Ray, “Design and FPGA prototype of 1024- bit Blum-Blum-Shub PRBG architecture,” in Proc. IEEE Int. Conf. Inf. Commun. Signal Process. (ICICSP), Singapore, Sep. 2018, pp. 38–43.
[16] T. Han and D. A. Carlson, “Fast area-efficient VLSI adders,” in Proc. IEEE 8th Symp. Comput. Arithmetic (ARITH), May 1987, pp. 49–56.
[17] D. L. Harris, “Parallel prefix networks that make tradeoffs between logic levels, fanout and wiring racks,” U.S. Patent 0 225 706 A1, Nov. 11, 2004.
[18] H. Ling, “High-speed binary adder,” IBM J. Res. Develop., vol. 25, no. 3, pp. 156–166, Mar. 1981.
[19] R. Jackson and S. Talwar, “High speed binary addition,” in Proc. Conf. Rec. 38th Asilomar Conf. Signals, Syst. Comput., vol. 2. Pacific Grove, CA, USA, Nov. 2004, pp. 1350–1353.
[20] K. S. Pandey, D. K. B. N. Goel, and H. Shrimali, “An ultra-fast parallel prefix adder,” in Proc. IEEE 26th Symp. Comput. Arithmetic (ARITH), Kyoto, Japan, Jun. 2019, pp. 125–134.
[21] F. Jafarzadehpour, A. S. Molahosseini, A. A. Emrani Zarandi, and L. Sousa, “New energy-efficient hybrid wide-operand adder architecture,” IET Circuits, Devices Syst., vol. 13, no. 8, pp. 1221–1231, Nov. 2019.
[22] S. Muthyala Sudhakar, K. P. Chidambaram, and E. E. Swartzlander, “Hybrid Han-Carlson adder,” in Proc. IEEE 55th Int. Midwest Symp. Circuits Syst. (MWSCAS), Boise, ID, USA, Aug. 2012, pp. 818–821.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY