iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.77835

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 III Mar 2026- Available at www.ijraset.com

Design of a Three-Operand Binary Adder Using Parallel Prefix Architectures

Dr. T. Madhavi Kumari[1], Chinnapally Muni Akhila[2]
Department of Electronics and Communication Jawaharlal Nehru Technological University Hyderabad (JNTUH) Hyderabad, India

Abstract: ***High-speed arithmetic operations are fundamental to modern digital systems, where multi-operand addition frequently determines the critical timing path. Conventional cascaded two-operand adders result in sequential carry propagation and increased latency. This paper presents an FPGA implementation employing carry computation to enhance timing performance. The proposed architecture decouples operand compression from carry resolution, thereby reducing critical path delay. Kogge–Stone, Han–Carlson, and Ladner–Fischer prefix structures are implemented in Verilog HDL and synthesized on an Artix-7 FPGA platform. Experimental results indicate that structured prefix networks improve computational efficiency while providing scalable trade-offs among delay, area, and power for high-throughput arithmetic systems.***
Keywords: Multi-operand binary adder, Parallel prefix structures, FPGA implementation.

I. INTRODUCTION

Arithmetic circuits play a crucial role in performing computations within modern digital systems such as processors, signal processing units, and embedded devices. Among fundamental arithmetic operations, binary addition is executed most frequently and often lies on the critical timing path. With increasing performance requirements in high-throughput computing environments, the design of adders with reduced latency and improved efficiency has become a primary focus in VLSI and FPGA-based system development.

Many real-time applications, such as multiply–accumulate units, partial product reduction in multipliers, and arithmetic logic unit (ALU) operations, require the addition of more than two operands within a single clock cycle. A conventional implementation performs this task using cascaded two-operand adders. However, such a sequential arrangement causes carry signals to propagate across multiple stages, thereby increasing the overall critical path delay. As operand width increases, this delay grows proportionally, limiting achievable clock frequency and degrading system performance.

To address these limitations, parallel prefix adders compute carry signals through structured and hierarchical logic networks. By leveraging associative carry operations, these architectures reduce carry propagation complexity to logarithmic depth with respect to operand width. Well-known prefix topologies, including Kogge–Stone, Han–Carlson, and Ladner–Fischer, are widely utilized in high-speed two-operand addition due to their efficient carry evaluation and systematic interconnection patterns. Nevertheless, existing research has predominantly focused on two-operand implementations, while comparatively fewer studies have explored structured three-operand addition combined with practical FPGA validation.

In this work, a structured three-operand binary adder is designed and realized on an FPGA platform using parallel prefix carry computation. The proposed architecture divides the operation into an intermediate compression stage followed by global carry resolution, thereby reducing sequential dependencies and shortening the critical path. Multiple prefix configurations—Kogge–Stone, Han–Carlson, and Ladner–Fischer—are integrated within a parameterized Verilog HDL framework. The designs are synthesized and implemented on an Artix-7 FPGA to evaluate delay, resource utilization, and power characteristics. The results provide a comparative hardware-level analysis and demonstrate the suitability of prefix-based multi-operand addition for scalable and high-performance arithmetic systems.

II. LITERATURE REVIEW

High-speed adder design has been widely studied in digital arithmetic because addition directly affects system performance. Early digital systems commonly used the Ripple Carry Adder (RCA) due to its simple structure and low hardware requirements. Ripple carry adders experience delay due to the step-by-step transfer of the carry signal. Since each stage depends on the carry generated by the previous stage, the overall delay increases as the number of bits grows. As a result, ripple architectures are not suitable for high-frequency or performance-critical systems.

To reduce delay, Carry Look-Ahead Adders (CLAs) were introduced. CLAs compute carry signals using generate and propagate logic, allowing partial parallel evaluation. This approach improves speed compared to ripple adders. However, the logic complexity and routing overhead increase significantly as the bit width grows. For large word sizes, this complexity limits scalability and increases area consumption.

Parallel prefix adders provide a more efficient solution for fast carry computation. These architectures use associative prefix operations to calculate carries through hierarchical logic stages. This structure reduces delay to logarithmic depth with respect to operand width. Among commonly used prefix topologies, the Kogge–Stone adder achieves minimal logic depth but requires extensive interconnections. The Han–Carlson architecture balances logic depth and wiring complexity through a hybrid structure. The Ladner–Fischer adder controls fan-out and interconnect overhead while maintaining efficient carry evaluation.

Most existing studies focus on optimizing two-operand prefix adders and comparing delay–area trade-offs across technologies. Several FPGA-based implementations have evaluated the timing and resource utilization of individual prefix structures described in Verilog HDL. Although these studies demonstrate improved speed over conventional adders, they primarily address two-operand addition.

In many practical systems, multi-operand addition is required in applications such as partial product reduction in multipliers and multiply–accumulate units. Traditional multi-operand implementations rely on cascaded two-operand adders, which introduce sequential carry propagation across stages and increase critical path delay. Structured three-operand addition using parallel prefix carry computation, combined with systematic FPGA-based comparison of multiple prefix topologies, remains relatively less explored.

Therefore, there is a need for scalable multi-operand prefix architectures with practical hardware validation. The present work addresses this need by integrating operand compression with hierarchical prefix carry computation and implementing multiple prefix configurations on an FPGA platform for comparative performance analysis.

III. TRADITIONAL PRACTICES

Conventional digital systems implement multi-operand addition using cascaded two-operand adders. Three input operands are processed sequentially: two operands first generate an intermediate result, which is then combined with the third operand. Although simple to design, this structure causes sequential carry propagation across stages.

In cascaded implementations, the final output depends on the completion of carry computation in each preceding stage. As a result, the overall critical path includes the delay of multiple adders. When the operand width increases, the accumulated carry propagation delay significantly affects timing performance. This limitation becomes more pronounced in high-speed applications where low latency and high operating frequency are required.

Moreover, the cascaded structure does not exploit parallelism in carry computation. Because every stage depends on the carry generated by the preceding stage, parallel computation of carry signals is not possible in this structure. This sequential dependency reduces scalability and makes the design inefficient for wide-word arithmetic operations.

Due to these limitations, conventional cascaded adders are not well-suited for performance-critical multi-operand arithmetic. These constraints motivate the adoption of structured parallel carry computation techniques that reduce carry dependencies and improve timing efficiency.

IV. PROPOSED METHODOLOGY

To address the delay limitations of cascaded multi-stage addition, a structured three-operand binary adder based on parallel prefix carry computation is developed. The proposed design performs operand compression at the bit level, followed by hierarchical carry evaluation using configurable prefix networks. This organization eliminates sequential carry dependency and enables scalable high-speed arithmetic implementation.

The methodology consists of three major stages: intermediate operand compression, prefix-based carry computation, and final sum generation.

A. Three-Operand Compression Stage

At each bit position i, three input bits A_i, B_i, and $C_i$are processed simultaneously. Instead of performing sequential two-operand additions, the proposed design locally compresses the three inputs into intermediate signals suitable for structured carry evaluation.

The intermediate signals are defined as:

$$Si' = Ai \oplus Bi \oplus Ci$$
$$Ci' = AiBi + BiCi + AiCi$$

Here, S_i' represents the preliminary sum bit, while C_i' represents the carry component generated at the bit position i. This compression stage reduces three operands into a format compatible with prefix-based carry propagation and operates independently at each bit position.

B. Parallel Prefix Carry Computation

After compression, global carry signals are computed using parallel prefix networks. Generate and propagate signals are derived from the intermediate values:

$$Gi = Ci'$$
$$Pi = Si'$$

Carry computation is performed using the associative prefix operator:

$$(Gk, Pk) \circ (Gj, Pj) = (Gk + PkGj, PkPj)$$

Due to its associative property, this operator allows hierarchical combination of carry signals across multiple stages. Consequently, carry resolution depth increases logarithmically with operand width, significantly reducing critical path delay.

To evaluate architectural trade-offs, three established prefix topologies are integrated within the proposed framework.

1) Kogge–Stone Topology

The Kogge–Stone structure achieves minimal logic depth by aggressively expanding prefix nodes at each stage. This configuration provides fast carry computation but increases interconnection density.

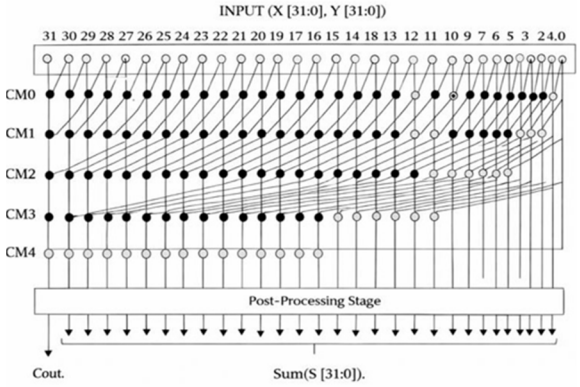


Fig. 1. Kogge–Stone parallel prefix carry structure

2) Han–Carlson Topology

The Han–Carlson prefix adder utilizes a mixed design approach by integrating both sparse and full prefix computation stages.This approach balances logic depth and wiring complexity, resulting in improved resource efficiency compared to fully expanded structures.

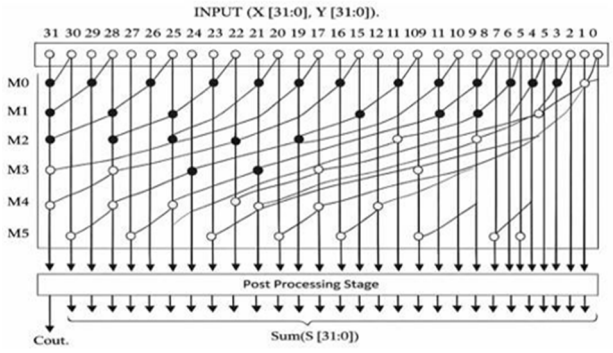


Fig. 2. Han–Carlson hybrid prefix carry structure

3) Ladner–Fischer Topology

The Ladner–Fischer structure offers flexible prefix tree construction with controlled fan-out. This topology reduces routing congestion while preserving logarithmic delay characteristics.

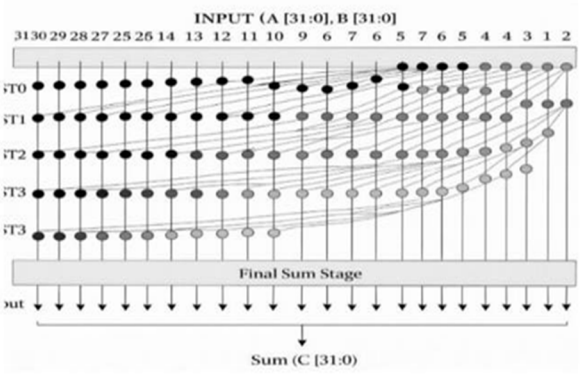


Fig. 3. Ladner–Fischer prefix carry structure

C. Three-Operand Adder Architecture

The complete architecture integrates the compression stage with the selected prefix carry network and final sum generation logic. After hierarchical carry signals C_i are resolved through the prefix structure, the final sum bits are computed as:

$$Si = Si' \oplus Ci$$

The final carry-out is obtained from the most significant prefix stage.

This integrated organization enables concurrent carry evaluation across all bit positions and eliminates the multi-stage dependency observed in conventional cascaded implementations. By combining operand compression with structured prefix computation, the proposed architecture achieves improved timing scalability for wide-word arithmetic systems.

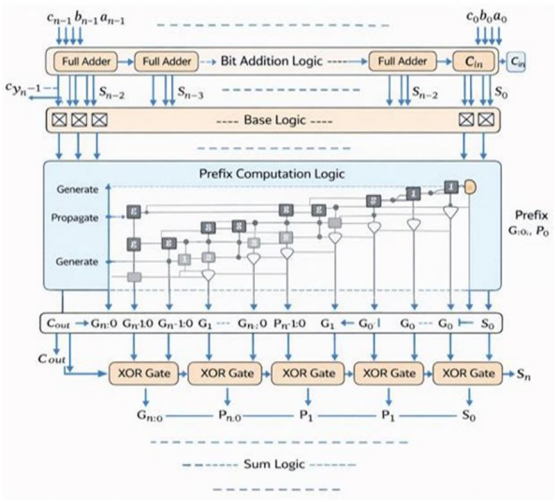


Fig. 4. Architecture of the proposed three-operand adder

D. FPGA-Based Hardware Realization

The proposed three-operand adder was designed and described using Verilog HDL. Independent parameterized modules were developed for the Kogge–Stone, Han–Carlson, and Ladner–Fischer prefix configurations. The operand compression logic and prefix network were structured as modular blocks to support scalability across different word lengths.

All designs were synthesized and implemented using the Xilinx Vivado toolchain targeting an Artix-7 FPGA device. Identical synthesis constraints were applied to each architecture to enable an unbiased comparison. Implementations were carried out for both 16-bit and 32-bit configurations.

Hardware utilization (LUTs and FFs), critical path delay, maximum operating frequency, and power consumption were obtained from post-synthesis and post-implementation reports in Vivado.

All evaluations were conducted under consistent operating conditions to ensure reliable architectural comparison.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

V. RESULTS AND DISCUSSION

The proposed three-operand parallel prefix adder architectures were synthesized and implemented on an Artix-7 FPGA device to evaluate hardware utilization, timing performance, and power consumption. All architectures were analyzed under identical synthesis constraints to ensure a fair and consistent comparison. Performance was evaluated for both 16-bit and 32-bit configurations.

A. Performance Evaluation for 16-Bit Architecture

The 16-bit implementations of the proposed three-operand prefix architectures were synthesized on the target FPGA device to evaluate area, delay, and power characteristics. Performance metrics were extracted from post-synthesis and post-implementation reports under identical design constraints. The resulting comparison is summarized in Table 1

Table 1: Performance Comparision Of 16-Bit Architectures

ADDER	AREA	DELAY	POWER
KOGGE-STONE	175	2.902	0.127
HAN-CARLSON	149	3.061	0.103
LADNER-FISCHNER	184	3.886	0.018

For the 16-bit implementation, the Kogge–Stone architecture achieves the lowest propagation delay (2.902 ns), indicating superior speed performance. This improvement is attributed to its fully expanded prefix network, which minimizes logic depth during carry computation. However, this advantage comes with increased area and power consumption compared to the other topologies.

The Han–Carlson architecture requires the fewest LUTs (149) and demonstrates significantly lower power consumption (0.018 W), making it more area- and energy-efficient. Although its delay is slightly higher than that of Kogge–Stone, it maintains competitive timing performance.

The Ladner–Fischer structure exhibits moderate area usage but higher delay relative to the other architectures. This behavior results from its balanced tree expansion, which reduces wiring complexity at the cost of increased prefix depth in certain stages.

B. Performance Evaluation for 32-Bit Architecture

To evaluate scalability, the proposed architectures were further implemented for a 32-bit operand width under the same synthesis constraints. Post-implementation reports were analyzed to examine variations in area utilization, timing performance, and power consumption with increased word length. The comparative results are presented in Table 2.

Table 2: Performance Comparison of 32-Bit Architectures

ADDER	AREA	DELAY	POWER
KOGGE-STONE	248	3.559	0.130
HAN-CARLSON	178	4.253	0.125
LADNER-FISCHNER	213	6.386	0.115

C. Comparative Analysis

The experimental results reveal clear architectural trade-offs among the evaluated prefix networks:

1) Kogge–Stone provides the highest speed performance but incurs higher area and power consumption.
2) Han–Carlson offers the best balance between delay, area, and power efficiency.
3) Ladner–Fischer achieves moderate area usage but exhibits higher delay in wider implementations.

Overall, integrating three-operand compression with structured parallel prefix carry computation enables scalable multi-operand addition with controlled timing growth. The FPGA results demonstrate that prefix-based architectures significantly improve performance compared to conventional cascaded multi-stage adders, particularly for wide-word arithmetic applications.

VI. CONCLUSION

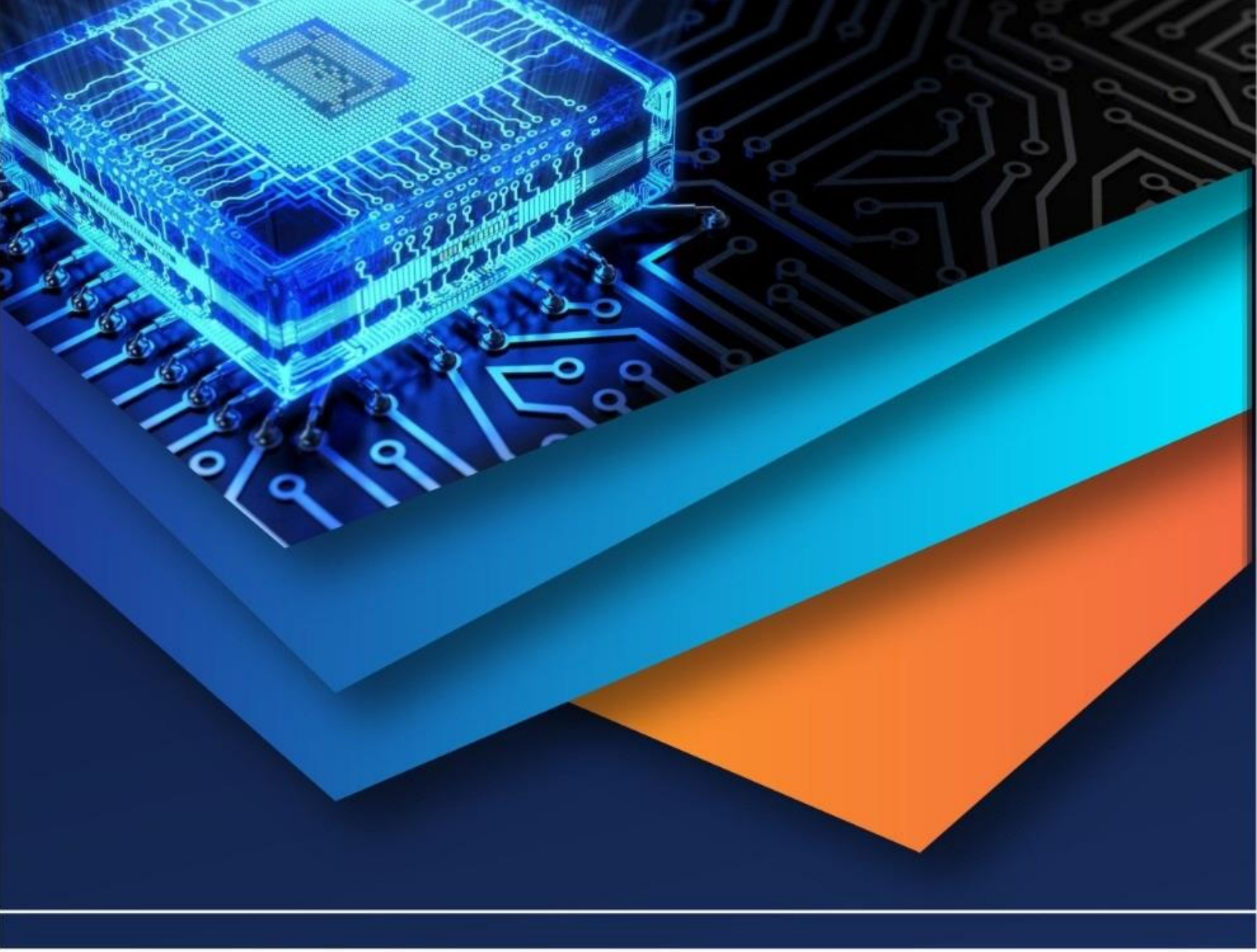
This work presented the design and FPGA implementation of a structured three-operand binary adder based on parallel prefix carry computation. The proposed architecture integrates bit-level operand compression with hierarchical carry resolution, thereby eliminating sequential carry dependency present in conventional cascaded designs. Three established prefix topologies—Kogge–Stone, Han–Carlson, and Ladner–Fischer—were implemented and evaluated on an Artix-7 FPGA platform for 16-bit and 32-bit configurations.

Experimental results demonstrate that the Kogge–Stone architecture achieves the lowest propagation delay, making it suitable for high-performance arithmetic applications. The Han–Carlson topology provides a balanced trade-off among area, delay, and power consumption, while the Ladner–Fischer structure offers moderate area efficiency with comparatively higher delay at larger operand widths. The observed timing trends confirm the logarithmic carry resolution property of prefix-based architectures, ensuring controlled delay growth with increasing word length.

Overall, the integration of three-operand compression with structured parallel prefix networks enables scalable and efficient multi-operand addition on FPGA platforms. The proposed framework offers a practical solution for high-throughput arithmetic systems requiring improved timing performance and architectural flexibility.

REFERENCES

[1] P. M. Kogge and H. S. Stone, “Parallel algorithms for efficient evaluation of recurrence relations,” IEEE Trans. Computers, vol. 22, no. 8, pp. 786–793, Aug. 1973.
[2] R. P. Brent and H. T. Kung, “A structured layout approach for parallel adder design,” IEEE Trans. Computers, vol. 31, no. 3, pp. 260–264, Mar. 1982.
[3] J. Ladner and M. J. Fischer, “Prefix computation techniques for parallel processing,” J. ACM, vol. 27, no. 4, pp. 831–838, Oct. 1980.
[4] S. Knowles, “Design considerations for a family of parallel adders,” in Proc. IEEE Symp. Computer Arithmetic, 2001, pp. 277–281.
[5] R. Zimmermann, “Efficient binary adder architectures for VLSI systems and synthesis,” Integration, VLSI J., vol. 28, no. 2, pp. 171–200, 1999.
[6] M. D. Ercegovac and T. Lang, Digital Arithmetic: Architectures and Implementations. Los Alamitos, CA, USA: IEEE Computer Society Press, 2004.
[7] Y. He and C. H. Chang, “Hybrid carry-lookahead and carry-select adder with improved power-delay performance,” IEEE Trans. Circuits Syst. I, vol. 51, no. 5, pp. 901–909, May 2004.
[8] B. Ramkumar and H. M. Kittur, “Low-power and compact carry select adder design,” IEEE Trans. VLSI Syst., vol. 20, no. 2, pp. 371–375, Feb. 2012.
[9] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-power CMOS digital circuit techniques,” IEEE J. Solid-State Circuits, vol. 27, no. 4, pp. 473–484, Apr. 1992
[10] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York, NY, USA: John Wiley & Sons, 1999.
[11] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, “Design of arithmetic logic units in sub-32 nm technologies,” IEEE J. Solid-State Circuits, vol. 44, no. 4, pp. 1268–1277, Apr. 2009.
[12] N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA, USA: Pearson Education, 2011.
[13] D. Harris and S. Harris, Digital Design and Computer Architecture, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2012.
[14] A. Morgenshtein, A. Fish, and I. A. Wagner, “Gate diffusion input logic for low-power digital circuits,” IEEE Trans. VLSI Syst., vol. 10, no. 5, pp. 566–581, Oct. 2002.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY