International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME123
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Design and Synthesis of 64-Bit Vedic Multiplier using Kogge-Stone Prefix Adder

Ms. K. Vasantha Laxmi[1], M Umamaheswari[2], N Riyaz Shaik [3], C Sai Prathyusha Reddy[4], G Mohith Kumar[5]

[1]*M. Tech, ECE Department, Annamacharya Institute of Technology and Sciences, Tirupati, vasanthalaxmik1998@gmail.com*

[2]*B. Tech, ECE Department, Annamacharya Institute of Technology and Sciences, Tirupati, umamuthyala71@gmail.com*

[3]*B. Tech, ECE Department, Annamacharya Institute of Technology and Sciences, Tirupati, riyazshaik53002@gmail.com*

[4]*B. Tech, ECE Department, Annamacharya Institute of Technology and Sciences, Tirupati, chithamreddyprathyusha@gmail.com*

[5]*B. Tech, ECE Department, Annamacharya Institute of Technology and Sciences, Tirupati, gettipatimohithkumar@gmail.com*

Abstract - A high-speed 64-bit Vedic multiplier based on the Urdhva Tiryakbhyam algorithm is presented in this work. The multiplier is implemented using a hierarchical architecture, where 2, 4 , 8, 16, and 32-bit multiplier blocks are recursively combined to construct the 64-bit design, enabling modularity and efficient reuse of hardware. Carry Save Adders (CSAs) are employed for intermediate partial-product accumulation to reduce carry propagation delay. A Kogge–Stone prefix adder is used as the final adder stage to achieve fast carry computation and high throughput. The combined use of Vedic multiplication, CSA-based reduction, and prefix addition results in improved speed performance, making the proposed design suitable for high-performance arithmetic and digital signal processing applications.

Keywords: Urdhva Tiryakbhyam, Hierarchical architecture, carry save adder (CSA), Kogge-Stone adder (Prefix adder), High speed.

1.INTRODUCTION

In present-day digital systems, arithmetic units play a vital role in determining the overall performance of processors, digital signal processing (DSP) units, and communication hardware. Among various arithmetic operations, multiplication is one of the most computationally intensive tasks, requiring more time and hardware resources than basic operations such as addition and subtraction. Efficient implementation of multiplication is therefore essential for applications including image processing, cryptography, artificial intelligence, and high-speed computing systems. As modern applications demand faster data processing, designing high-performance multipliers for large bit widths, such as 64-bit, has become increasingly important in VLSI design.

Conventional multiplier architectures, including array multipliers, Booth multipliers, and Wallace tree multipliers, are widely used in digital circuits. Although these methods are straightforward to implement, their performance degrades as the operand size increases due to longer carry propagation paths and sequential partial product accumulation. In high-bit-width operations, the addition stage contributes significantly to overall delay, limiting the speed of these traditional designs. Hence, there is a need for more efficient multiplier architectures that can achieve higher speed without excessive hardware complexity.

Vedic Mathematics offers an alternative approach for designing fast multipliers through algorithms that support parallel computation. The Urdhva-Tiryakbhyam technique, also known as the vertical and crosswise method, enables simultaneous generation of partial products, thereby reducing critical path delay. Additionally, its modular structure allows the construction of large multipliers by combining smaller
building blocks, making it well-suited for scalable VLSI implementations.
Despite the advantages of fast partial product generation, the overall performance of a multiplier also depends on the

efficiency of the adder used for accumulation. Conventional adders such as Ripple Carry Adders suffer from significant delay due to sequential carry propagation. In contrast, parallel prefix adders compute carry signals in a more efficient manner. Among them, the Kogge–Stone adder is particularly effective for large bit-width operations due to its logarithmic delay and high-speed performance.

In this work, a high-speed 64-bit Vedic multiplier is developed by integrating the Urdhva-Tiryakbhyam algorithm with efficient addition techniques. Carry Save Adders (CSA) are utilized to reduce intermediate carry propagation during partial product accumulation, while a 128-bit Kogge–Stone prefix adder is employed in the final stage to achieve fast computation. The proposed design focuses on improving speed and scalability while maintaining efficient hardware utilization, making it suitable for high-performance VLSI and DSP applications.

2.1 LITERATURE SURVEY

The design of high-speed multipliers is an important area in VLSI and digital signal processing systems, since multiplication is a fundamental operation in processors, ALUs, and DSP architectures. Various researchers have proposed different multiplier architectures to improve speed, reduce delay, and minimize power consumption. The Vedic multiplier, based on the Urdhva-Tiryakbhyam algorithm, has gained significant attention due to its parallelism and reduced computational complexity.

[1] S. S. Anu Thomas et al. compared Vedic, array, and Wallace tree multipliers. The results showed that the Vedic multiplier has lower delay and better hardware utilization due to its parallel structure, making it suitable for high-speed applications.

[2] D. Basha et al. proposed a Vedic multiplier using Ripple Carry Adder (RCA) and Carry Save Adder (CSA). The use of CSA reduces carry propagation delay in intermediate stages, which improves the overall speed of the multiplier.

[3] M. B. Murugesh et al. designed a modified high-speed 32-bit Vedic multiplier using optimized partial-product generation and efficient adders. Their FPGA implementation showed reduced delay and better performance compared to conventional designs.

[4] P. Gupta et al. discussed optimized transistor-level circuit techniques for improving speed and stability in VLSI systems. Such optimizations are useful for designing high-performance arithmetic circuits including multipliers.

[5] J. Zidar et al. presented dynamic voltage and frequency scaling (DVFS) to reduce power consumption in embedded systems. Power optimization methods are important for modern multiplier architectures used in low-power VLSI designs.

[6] H. Bhardwaj et al. proposed an improved interconnect delay model for deep submicron technology. Their work shows that delay in interconnections affects high-speed circuits, so efficient structures are required for adders and multipliers.

[7] D. B. Alaspure et al. developed a parallel Vedic processing architecture using ASIC design methodology. The parallel nature of the Vedic algorithm helps in increasing throughput and reducing computation time.

[8] H. Gupta and A. Kumar implemented a Vedic multiplier on FPGA and compared it with ripple carry and array multipliers. The results proved that the Vedic multiplier provides better speed and performance for digital systems.

[9] M. S. Babu and P. S. R. V. Prasad combined Booth encoding with the Vedic multiplier to reduce the number of partial products. This hybrid approach improves speed and reduces hardware complexity.

[10] R. Verma and P. Jain presented a comparison between Vedic and conventional multipliers for ALU applications. Their study concluded that Vedic multipliers provide better performance for higher bit operations.

2.2 EXISTING METHOD

Conventional multiplier designs used in digital systems are mainly based on array multipliers, Booth multipliers, and Wallace tree multipliers, which generate partial products sequentially and then combine them using basic addition methods. As discussed in [9], these traditional multiplier architectures work efficiently for small bit-width operations, but when the bit size increases to 32-bit or 64-bit, the number of partial products increases significantly, resulting in large propagation delay, higher power consumption, and increased hardware complexity. This limits their performance in high-speed VLSI and signal-processing applications.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME123
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Some existing implementations use array-based multiplication structures, where partial products are generated using AND operations and summed using ripple carry adders. As highlighted in [10], this approach is simple to design but suffers from long carry propagation delay, making it unsuitable for high-performance large-bit multipliers. These designs mainly focus on straightforward hardware implementation and do not consider the need for faster computation in modern processors.

In other existing approaches, Booth multipliers are used to reduce the number of partial products by encoding the multiplier bits. Although this method reduces the number of operations, the encoding and decoding logic increases circuit complexity. This leads to higher design overhead and makes the implementation difficult for large word-length multipliers.

Furthermore, several multiplier designs rely on non-modular structures, which makes scaling to higher bit widths difficult. These methods lack a proper hierarchical organization, resulting in poor scalability and reduced efficiency for 64-bit operations. Therefore, existing multiplier methods generally fail to provide a high-speed, scalable, and efficient architecture suitable for large-bit VLSI multiplication systems.

2.3 PROPOSED METHOD

The proposed 64-bit multiplier architecture integrates three major components:

A.Vedic partial product generation using the Urdhva Tiryakbhyam algorithm, B. Carry Save Adder (CSA) based partial product reduction, and C.A 128-bit Kogge–Stone parallel prefix adder for final summation.

The methodology follows a modular and hierarchical approach to improve scalability and simplify hardware implementation.

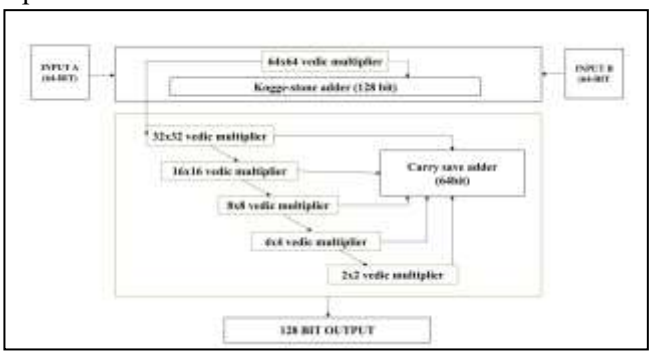


Fig -1: Block Diagram

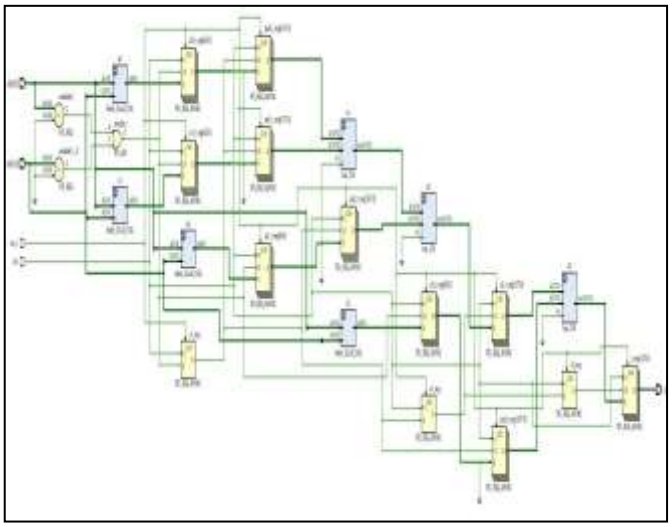


Fig -2: Schematic diagram of 64-bit vedic multiplier

The proposed 64-bit multiplier accepts two 64-bit inputs:

A = A [63:0], B = B [63:0]

And produces a 128-bit output:

P = P [127:0]

A.Partial Product Generation Using Vedic Multiplier:

The urdhva Tiryakbhyam (Vertically and Crosswise) algorithm is employed to generate the partial products for the 64-bit operands. For two n-bit inputs A=A[63:0] and B=B[63:0], the partial product at position i is computed as:

$$P_i=\sum_{k=0}^{i} a_k\, b_{i-k}$$

The key characteristics of this stage are:

1.Parallelism: All partial products are computed concurrently using AND gates.

2.Hierarchical Construction: The 64-bit multiplier is divided into smaller vedic blocks such as 2-bit,4-bit,8-bit,16-bit, 32-bit to simplify layout and improve scalability.

B. Partial Product Reduction Using Carry Save Adder:

After Vedic partial product generation, multiple partial product rows must be accumulated to obtain the final product. Direct addition using ripple carry adders leads to large carry propagation delay. Therefore, a Carry Save Adder (CSA) tree is employed for efficient multi-operand reduction.

A Carry Save Adder reduces three input operands into two outputs (Sum and Carry) without propagating carry across bit positions.

For three input bits at position i:

$$s_i=X_i \oplus Y_i \oplus Z_i$$

$$C_{i+1} = (X_iY_i) + (Y_iZ_i) + (X_iZ_i)$$

where:

- S_i= sum output
- C_{i+1}= carry output

The carry is shifted left by one position and added in the next stage.

C. 128-Bit Kogge–Stone Adder for Final Addition

After the partial product reduction using the Carry Save Adder (CSA) tree, the intermediate result is represented by two 128-bit vectors: the sum vector $P_{sum}$and the carry vector P_{carry}. These two vectors are finally added using a 128-bit parallel prefix adder, specifically the Kogge–Stone Adder, to obtain the final product.

1. Pre-Processing Stage

For two 128-bit input operands $A$and B, the generate and propagate signals at each bit position $i$are defined as:

$$G_i = A_i \cdot B_i$$
$$P_i = A_i \oplus B_i$$

where:

- $G_i$represents carry generation
- $P_i$represents carry propagation

2. Prefix Carry Computation

The carry computation follows the recursive relation:

$$C_{i+1} = G_i + (P_i \cdot C_i)$$

To compute carries in parallel, the Kogge–Stone adder uses a prefix operator defined as:

$$(G, P) \circ (G', P') = (G + P \cdot G',\ P \cdot P')$$

This operator combines adjacent generate–propagate pairs to form group generate and propagate signals. The prefix network computes carries in $\log_2 N$stages.

For a 128-bit adder:

$$\text{Logic depth} = \log_2(128) = 7$$

Thus, carry computation is achieved in only 7 stages.

3. Post-Processing Stage

Once the carries are computed, the final sum bits are obtained as:

$$S_i = P_i \oplus C_i$$

The final output is a 128-bit result.

2.4 RESULTS

The proposed 64-bit Vedic Multiplier was modeled using Verilog HDL and functionally verified through simulation. The simulation results confirm correct multiplication behavior for a wide range of operand combinations, including small values, large values, and boundary test cases. The output product (y[127:0]) consistently matched the mathematically expected results, demonstrating the correctness of the implemented Vedic multiplication algorithm.

The correctness of the output verifies the effectiveness of the Urdhva–Tiryagbhyam based multiplication scheme employed in the design. The inherent parallelism of the Vedic algorithm enables concurrent partial product computation, thereby reducing computational latency. This confirms that the proposed architecture is suitable for high-speed arithmetic applications requiring large-bit-width multiplication. The results demonstrate that the design maintains functional accuracy while supporting large operand sizes, making it appropriate for DSP, cryptographic, and high-performance computing systems.

Fig -3: Simulation Result

GRAPHS AND TABLES:

Fig -4: Resource Utilization Table

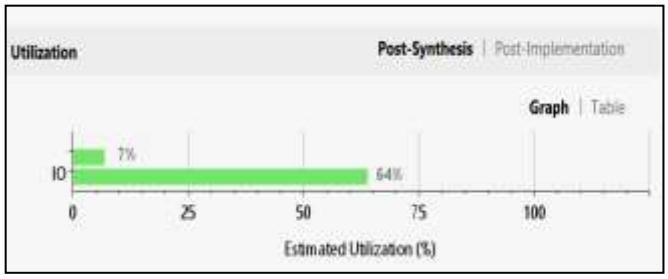
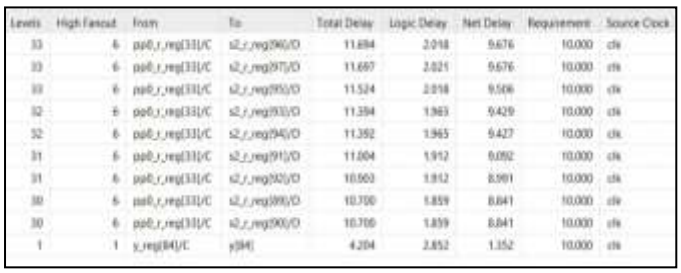


Fig -5: Resource Utilization Graph

Table -1: Performance comparison Table

Multiplier type	Delay (ns)	Area	Speed
Array multiplier	50-80	High	Slow
Wallace Tree	15-25	Medium	Moderate
Vedic+Ripple Carry Adder	20-30	Medium	Moderate
Vedic+CSA+ KSA	10-12	Slightly Higher	Fastest



Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
33	6	pp0_r_reg[33]/C	s2_r_reg[96]/D	11.694	2.018	9.676	10.000	clk
33	6	pp0_r_reg[33]/C	s2_r_reg[97]/D	11.697	2.021	9.676	10.000	clk
33	6	pp0_r_reg[33]/C	s2_r_reg[95]/D	11.524	2.018	9.506	10.000	clk
32	6	pp0_r_reg[33]/C	s2_r_reg[93]/D	11.394	1.965	9.429	10.000	clk
32	6	pp0_r_reg[33]/C	s2_r_reg[94]/D	11.392	1.965	9.427	10.000	clk
31	6	pp0_r_reg[33]/C	s2_r_reg[91]/D	11.004	1.912	9.092	10.000	clk
31	6	pp0_r_reg[33]/C	s2_r_reg[92]/D	10.903	1.912	8.991	10.000	clk
30	6	pp0_r_reg[33]/C	s2_r_reg[89]/D	10.700	1.859	8.841	10.000	clk
30	6	pp0_r_reg[33]/C	s2_r_reg[90]/D	10.700	1.859	8.841	10.000	clk
1	1	y_reg[84]/C	y[84]	4.204	2.852	1.352	10.000	clk

Fig -6: Delay

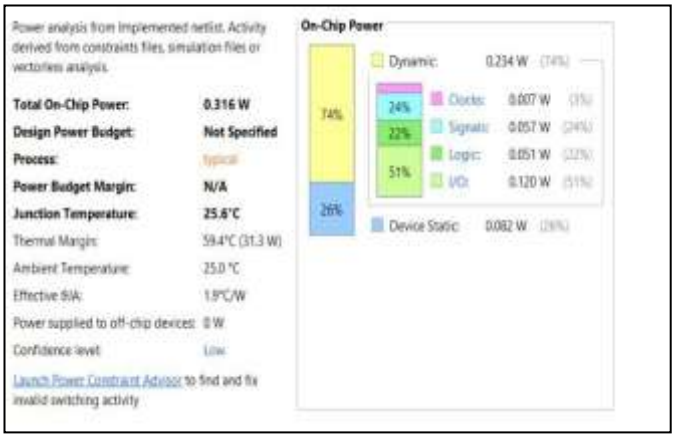


Fig -7: Power Consumption

3.CONCLUSIONS:

This paper presented the design and verification of a high-performance 64-bit Vedic Multiplier using the Urdhva–Tiryagbhyam algorithm with a Carry Save Adder (CSA) and Kogge–Stone Adder (KSA). The Vedic method generates partial products in parallel, the CSA reduces addition delay by minimizing carry propagation, and the Kogge–Stone adder performs fast final summation using parallel prefix computation. Simulation results confirmed correct 128-bit outputs for different input combinations, and the proposed architecture showed reduced delay and improved speed compared to conventional multipliers. Due to its modular and scalable structure, the design is suitable for high-speed applications such as DSP, cryptography, and high-performance VLSI systems.

FUTURE SCOPE:

The presented 64-bit Vedic multiplier using CSA–KSA can be further improved by applying advanced fabrication technologies to enhance speed and reduce power usage. Future enhancements may explore different prefix adder structures to balance performance and hardware cost. Additionally, the design can be scaled to larger bit widths such as 128-bit and used in high-speed digital and signal processing systems.

RERERENCES

[1]. S. S. Anu Thomas, Ashly Jacob and S. Sudhakaran, “Comparison of vedic multiplier with conventional array and wallace tree multiplier,” International Journal of VLSI System Design and Communication Systems, vol. 04, pp. 244–248, 04 2016.

[2]. D. Basha et al., “Rca-csa adder based vedic multiplier,” International Journal of Applied Engineering Research, vol. 12, pp. 7603–7613, 01 2017.

[3] M. B. Murugesh et al., “Modified high speed 32-bit vedic multiplier design and implementation,” in 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 929–932.

[4]. P. Gupta, N. Singh, V. Kumar, and K. Kumar, “Implementation of Dual Node 12T SRAM Unit Cell,” 2024. doi: 10.1109/IC2E362166.2024.10827406.

[5]. J. Zidar et al., “Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems,” Electronics, vol. 13, p. 826, 02 2024.

[6]. H. Bhardwaj, S. Jain, and H. Sohal, “An innovative interconnect structure with improved elmore delay estimation model for deep submicron technology,” Analog Integrated Circuits and Signal Processing, vol. 111, pp. 1–21, 06 2022.

[7]. D. B. Alaspure, S. R. Dixit and J. S. Edle, “Design and Development of Parallel Vedic Processing Architecture through ASIC Design Methodology,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 10S, pp. 477–486, 2024.

[8].H.Gupta and A.Kumar, “Vedic Multiplier Implementation on FPGA: A Comparative Study with Ripple Carry and Array Multipliers,” Int. J. Comput. Appl., vol. 183, no. 30, pp. 19–27, Mar. 2023.

[9] M. S. Babu and P. S. R. V. Prasad, “High-Speed Vedic Multiplier Design Using Booth Encoding for DSP Applications,” Int. J. Eng. Sci. Technol., vol. 13, no. 7, pp. 558–565, Jul. 2021.

[10]. R.Verma and P. Jain, “Comparative Study of Vedic and Conventional Multiplication Techniques for Arithmetic Logic Units,” Int. J. Eng. Innov. Technol., vol. 14, no. 4, pp. 103–111, Jun. 2025.