32-Bit Design and Implementation of an Efficient Modified Vedic Multiplier.

Suroju Sai Santhosh Kumar
Department of ECE
Vaagdevi College of Engineering
Telangana, India.
saisanthosh18@gmail.com

Udutha Saritha
Department of ECE
Vaagdevi College of Engineering
Telangana, India.
Saritha_u@vaagdevi.edu.in

***Abstract* - This paper presents the design and implementation of a high-speed, area-efficient 32-bit Vedic Multiplier. The architecture leverages the ancient Urdhva Tiryakbhyam Sutra for parallel partial product generation through a detailed hierarchical implementation from the 2-bit gate level up to the 32-bit top level. Existing modified Vedic multipliers have improved performance by using Ripple Carry Adders (RCAs), achieving a delay of 68.859 ns. To significantly enhance this performance, the proposed design incorporates a deeply recursive modular structure and utilizes Kogge-Stone parallel prefix adders (PPAs) for rapid intermediate sum accumulation. The design's operation, including bit-slicing and partial product summation, is detailed. Described in Verilog HDL and synthesized using Xilinx Vivado, the results show a datapath delay of 22.179 ns, marking a 65% reduction compared to the RCA-based model, making it highly suitable for high-performance DSP, ALU, and other VLSI-based systems.**

Keywords— Vedic Multiplier, Urdhva Tiryakbhyam, Kogge-Stone Adder, Ripple Carry Adder (RCA), Parallel Prefix Adder (PPA), VLSI, High-Speed Multiplier.

I. INTRODUCTION

Multiplication is a critical operation in digital arithmetic units, particularly in Arithmetic Logic Units (ALUs), Digital Signal Processors (DSPs), and embedded processors. Traditional architectures such as Booth and Wallace Tree multipliers achieve various trade-offs in terms of delay, area, and complexity. However, as modern applications increasingly demand high-speed and low-power hardware, there is a growing interest in mathematically efficient alternatives. One such promising approach is derived from Vedic mathematics.

Vedic mathematics, rooted in ancient Indian computation techniques, introduces compact and parallelizable methods for arithmetic operations. Among the sixteen sutras described, the Urdhva Tiryakbhyam (UT) Sutra—translated as "Vertically and Crosswise"—is particularly effective for binary multiplication. Unlike conventional array or sequential multipliers, this method allows for simultaneous generation of all partial products, inherently enabling a recursive and modular design

The UT-based multiplier decomposes large operands recursively. For instance, a 32×32-bit multiplication is broken down into four 16×16 multiplications, each of which is further decomposed down to 2×2 multipliers at the lowest level. This recursive decomposition supports scalable, pipelined architectures that suit FPGA and ASIC design flows.

In this work, a high-performance 32-bit Modified Vedic Multiplier architecture is proposed, which integrates the UT Sutra with Kogge-Stone Parallel Prefix Adders (KS-PPA) for partial product accumulation. Kogge-Stone Adders are known for their minimal logical depth and logarithmic carry propagation delay, making them ideal for high-speed arithmetic applications. By combining these two techniques, the proposed architecture achieves a significant reduction in datapath delay while maintaining regularity and modularity, making it highly suitable for VLSI synthesis and implementation in real-time systems.

This paper presents the full Verilog implementation of the multiplier, from the 2×2 base case to the top-level 32×32 module, optimized using modular KS-PPAs at each level. The recursive structure not only ensures reusability and efficient routing but also supports pipelining for higher throughput. Simulation and synthesis results reveal a datapath delay improvement of nearly 60% over traditional architecture.

II. LITERATURE SURVEY

Many Every circuit presented can be utilized in the most efficient manner possible. Quantum-dot Cellular Automata (QCA) has emerged as a viable nanoscale alternative to CMOS technology, offering higher density, speed, and reduced power consumption. The proposed designs further minimize the total footprint across various

circuit types, including processors and arithmetic logic units (ALUs) [1].

Kishore et al. [2] introduced a method to compute instantaneous convolution using Vedic mathematics. The implementation of the Vedic Urdhva Tiryagbhyam (UT) multiplier has proven effective in rapidly managing graphical convolution results and emulating both traditional and modern convolution solutions. Eshack and Krishnakumar [3] implemented a pipelined low-power Vedic multiplier. Pipelining, as a design strategy, is instrumental in reducing power consumption while preserving speed. Their use of pipelined

Gupta and Sharma [4] proposed high-speed Vedic multipliers utilizing the Han–Carlson adder structure. These designs are particularly suited for signal processing and transmission systems that require fast and energy-efficient computations.

Furthermore, advanced 32×32-bit Vedic multiplier architectures have been investigated, highlighting performance benefits in complex multiplication tasks and ease of implementation compared to conventional designs [4].Deepa and Marimuthu [4] also developed a novel multiplier architecture capable of handling positive, negative, and mixed-sign inputs.

For multiplier design, the Urdhva Tiryagbhyam Sutra remains the most efficient and computationally effective approach [5]. This Sutra differs from traditional multiplication methods by enabling all partial product generations in parallel, offering superior speed and accuracy. A Modified Carry Save Adder (MCSA) is incorporated to compute the sum of partial products, minimizing update latency. The implementation is described in Verilog HDL for effective hardware synthesis.

Akhter and Chaturvedi [6] proposed an innovative binary multiplier architecture grounded in Vedic mathematics. Their synthesis experiments confirm improved latency and LUT (Look-Up Table) usage efficiency. However, the study does not provide a comparative analysis of power consumption.

Prabhu et al. [7] conducted an extensive review of applicable Vedic sutras and evaluated the Booth algorithm for encoding and decoding tasks. Reddy et al. [8] proposed a practical 8×8 Vedic multiplier based on a modular 4×4 design.

Santhosh Kumar and Gowrishankar [9] integrated Vedic multipliers inspired by the Yavadunam Sutra across various stages of digital design. These include computation kernels, data reduction, storage compaction, and quantization, all crucial in managing the processing and storage demands of deep neural networks (DNNs).

Finally, Khubnani et al. [10] examined the cross-domain applicability of Vedic multiplication techniques, showcasing their relevance in multiple engineering fields and digital architectures.

III. METHODOLOGY

In To achieve high-throughput arithmetic operations, the proposed architecture follows a recursive approach to multiplication using the Vedic Urdhva Tiryakbhyam (UT) Sutra, enhanced with fast accumulation through Kogge-Stone Parallel Prefix Adders (PPAs). The multiplier structure is modular, where the 32-bit operation is constructed from smaller operand widths, breaking down progressively until the base 2×2 multiplier level.

Initially, the 32-bit input operands are divided into high and low 16-bit segments, denoted as A=AH‖ALA = A_H \Vert A_LA=AH‖AL and B=BH‖BLB = B_H \Vert B_LB=BH‖BL, where AH=A[31:16]A_H = A[31:16]AH=A[31:16], AL=A[15:0]A_L = A[15:0]AL=A[15:0], and similarly for operand BBB. These segments are used to compute four partial products simultaneously:

PP0=AL×BLPP_0 **-(1)**
PP1=AL×BHPP_1 **-(2)**
PP2=AH×BLPP_2 **-(3)**
PP3=AH×BHPP_3 **-(4)**

These products are generated in parallel using four instances of 16×16 Vedic multipliers. Each of these submodules recursively implements the same strategy, eventually invoking 2×2 multipliers at the base level. This recursion provides architectural uniformity and scalability.

After partial products are computed, they are aligned based on their significance in the final result. Specifically, **PP_1** and **PP_2** are left-shifted by 16 bits (multiplied by 2162^{16}216), while **PP_3** is shifted by 32 bits. These aligned partial results are then combined using a two-stage accumulation strategy employing Kogge-Stone adders.

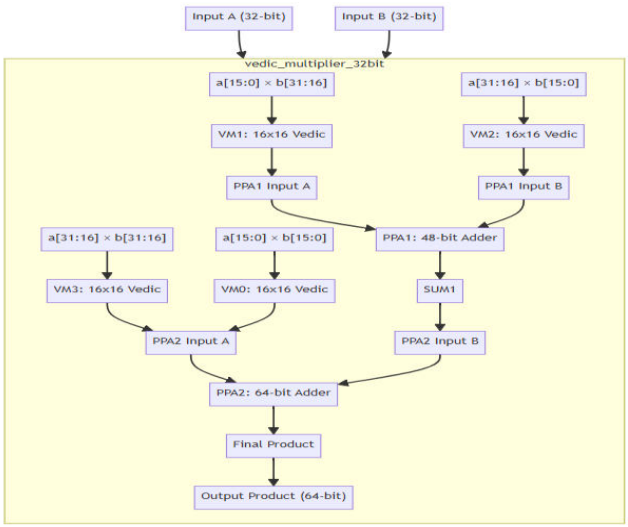


Fig:1 Proposed methodology.

In the first stage, the intermediate sum is computed as:

$$\mathbf{SUM1=(PP1\ll 16)+(PP2\ll 16)}$$

This operation is handled by a 48-bit Kogge-Stone adder to ensure high-speed carry propagation. The second stage computes the final product by summing $\mathbf{PP_0}$, the previously computed $\mathbf{SUM_1}$, and the shifted $\mathbf{PP_3}$:

$$\mathbf{PRODUCT=(PP3\ll 32)+PP0+(SUM1\ll 16)}$$

A 64-bit Kogge-Stone adder is used at this stage to manage the full-width summation and avoid overflow. The use of prefix adders at each accumulation level reduces the overall logic depth, allowing faster carry generation compared to ripple-carry or carry-select approaches.

This entire computation sequence ensures that the multiply-and-accumulate path remains fully combinational and parallel at each hierarchy, which is ideal for pipelined or clocked environments that demand high arithmetic throughput.

IV. IMPLEMENTATION

MODULE: A recursive Vedic multiplication algorithm combined with Kogge-Stone parallel prefix adders for efficient partial product summation. The 32-bit operands AAA and BBB are decomposed into their lower and upper 16-bit halves:

$$A = A_H \times 2^{16} + A_L, \quad B = B_H \times 2^{16} + B_L$$

Partial products from eq1-eq4 are combined to compute the full product PPP as:

$$P = pp_3 \times 2^{32} + (pp_1 + pp_2) \times 2^{16} + pp_0$$

Below schematic shows a Verilog implementation of 32 bit Vedic multiplication module.It shows how the ksa and the 2x2 ,4x4,16x16 modules are connected. A Verilog simulation program was used to create this schematic.

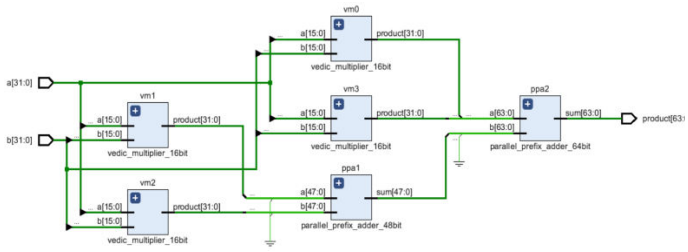


Figure 2: Schematic diagram in Verilog

The addition steps are accelerated using parallel prefix adders implemented via the Kogge-Stone algorithm, minimizing carry propagation delay.

This recursive decomposition continues at lower bit widths: each 16-bit multiplier splits into 8-bit multipliers, each 8-bit into 4-bit, and each 4-bit into 2-bit multipliers. At the 2-bit level, the base Vedic multiplier uses the Urdhva Tiryakbhyam sutra to perform multiplication via bitwise AND and half-adders, efficiently producing a 4-bit product.

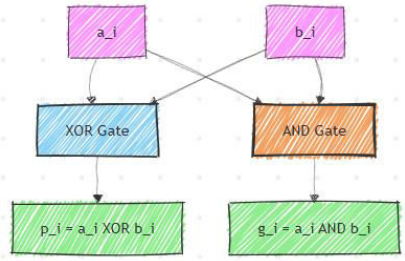


Fig:3 Propagate-Generate (PG) Generator

The parallel prefix adders (ranging from 6-bit to 64-bit) calculate propagate (ppp) and generate (ggg) signals for each bit:

$$p_i = a_i \oplus b_i, \quad g_i = a_i \cdot b_i$$

Carry signals are generated through multiple prefix levels:

$$G_{j:i} = G_{j:k} + P_{j:k} \cdot G_{k-1:i}$$

where PPP and GGG denote group propagate and generate, enabling rapid carry computation.

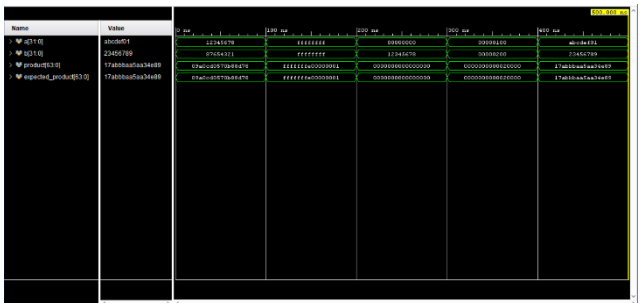
Add these partial products, the design employs Kogge-Stone parallel prefix adders, which use propagate (pip_ipi) and generate (gig_igi) signals derived from the input bits to perform carry computation in logarithmic time, avoiding the delay of ripple carry addition. In the first addition stage, the two middle partial products pp1pp_1pp1 and pp2pp_2pp2 both zero-padded to 48 bits are summed using a 48-bit parallel prefix adder. The output of this addition, sum1sum1sum1, also has a width of 48 bits. In the second stage, the highest partial product pp3pp_3pp3 concatenated with the lowest partial product pp0pp_0pp0 forms a 64-bit vector, which is then added to sum1sum1sum1 shifted left by 16 bits (again by zero concatenation) using a 64-bit parallel prefix adder. The final output from this adder, final_sumis a 64-bit wide vector representing the full product of the 32-bit inputs.

The Kogge-Stone adder internally generates the propagate and generate signals for every bit and computes carries through a tree of prefix operations that enable parallel carry propagation, drastically reducing addition latency. This results in fast and accurate summation of the aligned partial products. The final 64-bit product, after this hierarchical addition of shifted partial products, is assigned as the output of the 32-bit Vedic multiplier module. This structured approach of recursively generating partial products and summing them efficiently using parallel prefix adders and bit alignment through zero padding

enables the design to achieve high-speed multiplication with scalable hardware efficiency.

V. RESULTS AND DISCUSSION

The 32-bit Modified Vedic Multiplier with Parallel Prefix Adders was simulated to verify its functionality. The simulation results, as shown in the provided waveform, demonstrate the correct operation of the multiplier. For input values **A=0xabcdef01** and **B=0x23456789**, the observed product was **0x17abbbaa5aa34e89**. This perfectly matches the expected product, confirming the accuracy of the design. The waveform clearly illustrates the stability of the output product[63:0] at the calculated value after a certain propagation delay, validating the integrity of the hierarchical Vedic multiplication structure combined with the fast carry generation capabilities of the Kogge-Stone



parallel prefix adders used in the design.

Fig 3: 32-bitVedic multiplier KSA verilog waveform

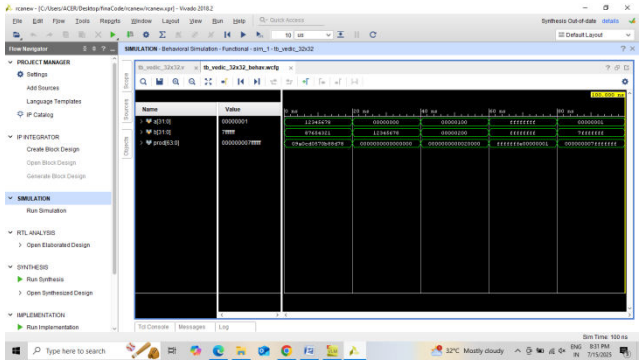


Fig 4: 32-bitVedic multiplier RCA verilog waveform

The simulation is from an HDL simulator running a Verilog testbench. The testbench indicatestesting a vedic multiplication module. The waveform above captures the signal transition across time, measured in nanoseconds (ns).

VI. Conclusion And Future Scope

The delay analysis provides a clear demonstration of the performance benefits of the proposed 32-bit Vedic multiplier with a Kogge-Stone Adder (KSA). When compared to a 32-bit Vedic multiplier employing a basic Ripple Carry Adder (RCA), which recorded a substantial delay of 68.859 ns, the proposed KSA-based design achieves a remarkable reduction to just 22.179 ns. This represents an improvement of approximately 67.6% in operational speed.

Multiplier	Delay(in ns)
32-bit Vedic with RCA	68.859ns
32-bit Vedic With RCA	30.854ns
32-bit Vedic with Proposed KSA	22.179ns

Fig: Delay comparison with existing designs

Furthermore, even against a more optimized 32-bit Vedic multiplier using an RCA (with a delay of 30.854 ns), the proposed KSA implementation still offers a significant advantage. The 22.179 ns delay of the proposed design translates to an approximate 28% speedup over this comparatively faster RCA version. These results unequivocally establish the superior performance of the 32-bit Vedic multiplier integrated with the proposed KSA, making it a highly efficient solution for high-speed arithmetic computations by effectively mitigating propagation delays inherent in traditional adder architectures.

LUT's Utilization presents a clear comparison of Look-Up Table (LUT) utilization across various Vedic multiplier architectures of different bit-widths. As expected, there is a general trend of increasing LUT usage with an increase in bit-width, reflecting the greater complexity of larger multipliers.

For 8-bit Vedic multipliers, the LUT utilization is minimal at 201. Scaling up to 16-bit, the standard Vedic multiplier uses 796 LUTs. A "16-bit modified Vedic" multiplier shows a slight increase to 903 LUTs, suggesting that the modifications introduce a small overhead in terms of logic resources.

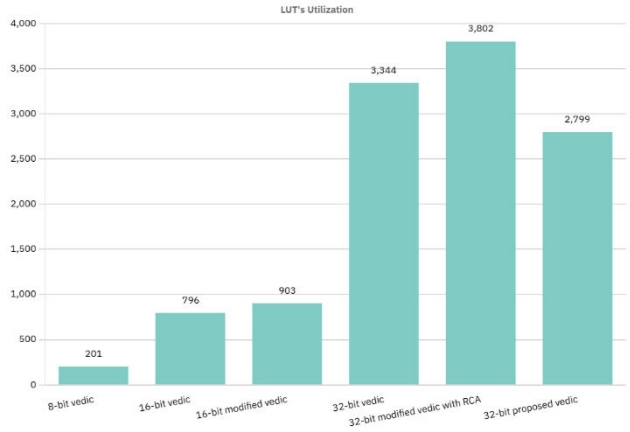


Fig:4 LUT's Utilization graph.

The trend continues with 32-bit multipliers. A standard "32-bit Vedic" multiplier consumes 3,344 LUTs. The "32-bit modified Vedic with RCA" (Ripple Carry Adder) exhibits the highest LUT utilization among all presented designs, reaching 3,802. This indicates that integrating a Ripple Carry Adder within the modified Vedic architecture at 32-bit significantly increases the hardware

resource consumption. In contrast, the "32-bit proposed Vedic" multiplier demonstrates a notable reduction in LUTs compared to the "32-bit modified Vedic with RCA", utilizing 2,799 LUTs. This suggests that the proposed modifications or architecture in the 32-bit design are more efficient in terms of area, requiring fewer LUTs than the RCA-integrated modified design, and even fewer than the standard 32-bit Vedic multiplier.

REFERENCES

1.Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda"pp. 5- 45, Motilal Banarasidas Publishers, Delhi, 2009.

2. Sai Venkatramana Prasada G S, G Seshikala, Niranjana S "Design of High Speed 32-bit Floating Point Multiplier using Urdhva Tiryagbhyam Sutra of Vedic Mathematics", International Journal of Recent Technology and Engineering (IJRTE), Vol. 8, Issue-2S3, July 2019.

3. Manpreet Kaur, Amandeep Kaur, "Design and Analysis of Vedic Multiplier by Using Modified Full Adders", International Journal of Engineering Development and Research (IJEDR), Vol. 6, Issue 2, 2018.

4. P. Manju, P.B. Sreelakshmi, T. Madhavi, "Design of Vedic Multiplier Using GDI method", International Journal of Recent Technology and Engineering (IJRTE), Vol. 7, Issue 6S4, April 2019.

5. Arunkumar P. Chavan, Rahul Verma, Nishanth S. Bhat "High Speed 32-bit Vedic Multiplier for DSP Applications", International Journal of Computer Applications, Vol.135- No.7, February 2016.

6. Siba Kumar Panda, Ritisnigdha Das, S K Saifur Raheman, Tapasa Ranjan Sahoo, "VLSI Implementation of Vedic Multiplier Using Urdhva Tiryagbhyam Sutra in VHDL Environment: A Novelty", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), Vol.5, Issue 1, pp 17-24, Jan-Feb 2015.

7. Syed Shahzad Hussain Shah, Muhammad Naseem Majoka, Gulistan Raja, "Design and Implementation of 32-bit Vedic Multiplier on FPGA", First International Conference on Modern communication & Computing Technologies, Research Gate, February 2014.

8. S. Nithya Devi, B. Gopinath, P. Ramanathan, "Synthesis of High Speed Vedic Multiplier", International Journal of Applied Engineering Research, Vol.10, No. 29, May 2015.

9. S. Jayakumar, R. Selvam, K.Karthikeyan, R. Natesan, "Design and Implementation Hybrid FIR Filters using Vedic Multiplier and Fast Adders", International Journal of Recent Technology and Engineering (IJRTE), Vol.8, Issue 4, November 2019.

10. Khushboo Pichhode, Mukesh D. Patil, Divya Shah, Chaurasiya Rohit B. "FPGA Implementation of Efficient Vedic Multiplier", International Conference on Information Processing, December 2015.

11. S. Akhter, "VHDL implementation of fast N×N multiplier based on Vedic mathematics," in Proc. 18th European Conference on Circuit Theory and Design, 2007, pp. 472-475.

12. B. D. Kumar and M. Bharathi, "A high speed and efficient design for binary number squaring using Dwandwa yoga," International Journal of Advanced Research in Computer Engineering & Technology, vol. 1, no. 4, pp. 476-479, Jun. 2012.

13. H. Goyal and S. Akhter, "VHDL implementation of fast multiplier based on Vedic mathematic using modified square root carry select adder," International Journal of Computer Applications, vol. 127, no. 2, pp. 24-27, Oct. 2015.

14.S. Akhter, V. Saini, and J. Saini, "Analysis of Vedic multiplier using various adder topologies," in Proc. IEEE 2017 International Conference on Signal Processing and Integrated Networks, 2017, pp. 173-176.

15. A. Deepa and C. N. Marimuthu, "High speed VLSI architecture for squaring binary numbers using Yavadunam sutra and bit reduction technique," International Journal of Applied Engineering Research, vol. 13, no. 6 , pp. 4471-4474, 2018.