AOI: 10.100.234513.0266
13th international conference on Microelectronics, Circuits and Systems, Micro2026

Implementation of an Optimized 16-Bit Vedic Multiplier Using Lilavati Adder on FPGA for High-Speed Digital Applications

Mili Sarkar, Devesh Kumar, Piyush Bhatt, Ujjwal Kant, and Riya Sai Nayak

Department of Electronics and Communication Engineering
Institute of Engineering and Management (IEM), Kolkata, WB, India
Corresponding Author: mili.sarkar@iem.edu.in

ABSTRACT

Multiplication is a basic arithmetic operation used in various digital signal processing systems, image processing systems, cryptographic systems, and high-speed computation systems. The performance of the above systems is highly dependent on the propagation delay of the multiplier architectures. This paper proposes an optimized 16×16 multiplier architecture to enhance the computation speed. The computation speed of the proposed multiplier is improved by optimizing the partial product generation. The proposed multiplier architecture is developed by using a hierarchical modular structure and a column-wise accumulation method by using the Lilavati addition method. The proposed design is coded using Verilog HDL and implemented on a Xilinx Artix-7 FPGA platform by using the Vivado design tools. The result obtained by implementing the proposed multiplier architecture on the Xilinx platform shows a propagation delay of 20.81 ns. The maximum frequency of the proposed multiplier is approximately 48 MHz. The propagation delay of the traditional multiplier architectures proposed by various authors using the same implementation platform is within the range of 23 ns to 80 ns. The proposed method outperforms the traditional method by achieving a lower propagation delay.

Keywords: Vedic Multiplier, Lilavati Addition (LA), Urdhva-Tiryagbhyam (UT), Population Counter, 16-to-5 Compressor.

I. INTRODUCTION

In the modern era of VLSI design, with the help of VLSI technology,the need for a high-speed and efficient computation blocks have increased exponentially. The multiplier block is considered an integral part of Digital Signal Processing, Microprocessors, and Cryptography [1, 5, 14]. The conventional multiplication algorithms, i.e., Array Multiplication and Booth Multiplication, have a major bottleneck problem due to Carry Propagation Delay (CPD) with the increased bit size of the multiplier [21]. To overcome the above challenges, this paper proposes a novel optimized design for a 16-bit multiplier based on the principles of Vedic Mathematics, which was used in the ancient era [11, 16, 17]. The proposed design is structured into three different layers: To address these challenges, this paper proposes an optimized 16-bit multiplier architecture based on ancient Vedic Mathematics combined with modern parallel accumulation techniques [10, 15] . The proposed design is structured into three distinct layers to maximize throughput:

Layer 1 - Partial Product Generation: This layer utilizes the **UT (Urdhva-Tiryagbhyam)** Sutra, which allows for the decomposition of a 16-bit multiplication into four parallel 8×8 modular blocks [10, 15]. The UT Sutra translates to "Vertically and Crosswise," enabling simultaneous generation of partial products and reducing the initial computational latency [11, 16].

Layer 2 - Parallel Accumulation: The novel part is introduced in the LA (Lilavati Addition) step. In this step, unlike traditional Ripple Carry Adder (RCA) architectures [12], the intermediate results are treated as a "Bit Forest." Each column is independently added using high-speed population counters like the 16-to-5 Compressor. This technique of independent addition of columns results in a reduced critical path delay because of the limited carry dependency between columns [18, 21].

Layer 3 - Normalization: The final stage employs a hard-wired positional alignment to align the weighted sums from the LA stage, followed by a shallow reduction tree to produce the final 32-bit product [13, 20].

The primary objective of this research is to demonstrate that the combination of UT-based decomposition and LA-based parallel compression significantly reduces the Power-Delay Product (PDP) compared to existing architectures [8, 9]. The remainder of this paper is organized as follows. Section II presents the mathematical formulation of the proposed architecture, Section III describes the proposed methodology, Section IV discusses the FPGA implementation and results, and Section V concludes the paper along with future work.

II. MATHEMATICAL FORMULATION

The proposed multiplier architecture is mathematically derived from the combination of the Urdhva-Tiryagbhyam (UT) sutra for partial product generation and the Lilavati Addition (LA) principle for column-wise reduction [11, 16].

A. Partial Product Generation (UT Sutra)

Let two 16-bit numbers be represented as A and B. In the modular decomposition phase, these inputs are divided into high and low 8-bit halves:

$$(A = A_H \cdot 2^8 + A_L), \quad (B = B_H \cdot 2^8 + B_L) \qquad (1)$$

ISBN: 978-81-689089-0-1

where A_H , A_L , B_H , $B_L \in [0, 2^8 - 1]$. The total product P is formulated as:

$$P = (A_H B_H)2^{16} + (A_H B_L + A_L B_H)2^8 + (A_L B_L) \quad (2)$$

This leads to four 16-bit partial product terms: P_{HH}, P_{HL}, P_{LH}, and P_{LL} [10, 15].

B. Bit Forest and Lilavati Accumulation

Unlike traditional binary addition, the LA stage treats these partial products as a matrix of bits $b_{i,j}$, where i denotes the column weight and j denotes the bit position in the vertical stack. The total product P can be expressed as the weighted sum of column-wise population counts:

$$P = \sum_{k=0}^{31} \left(\sum_{j=0}^{h_k - 1} b_{k,j} \right) \cdot 2^k \quad (3)$$

where h_k is the bit density (height) of the k-th column.

C. 16-to-5 Compressor Logic

For the maximum density columns where $h_k = 16$, a specialized population counter (compressor) is used. The output of the compressor S_k is a 5-bit vector representing the arithmetic sum of logic '1's in that column:

$$S_k[4:0] = Count\left(\sum_{j=0}^{15} b_{k,j} \right) \quad (4)$$

This summation is performed in a purely vertical manner, ensuring that:

$$\frac{\partial S_k}{\partial S_{k-1}} = 0 \quad (5)$$

The partial derivative above proves that the result of column k is independent of column k − 1, mathematically validating the Zero Carry Propagation Delay [18, 21].

D. Final Normalization

The final product is computed by merging the shifted outputs of the LA stage:

$$P = \sum_{k=0}^{31} S_k \cdot 2^k \quad (6)$$

This is implemented using a hard-wired barrel shifter for the 2^k scaling and a shallow reduction tree of Half Adders and Full Adders for the final merge [13, 20].

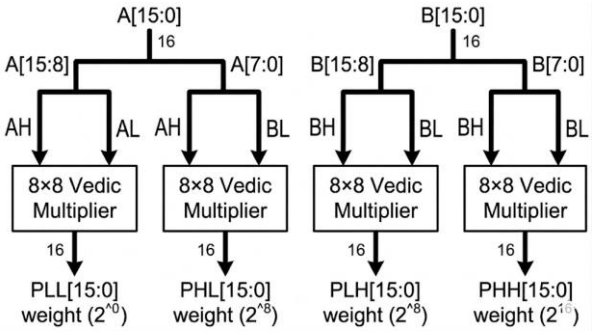

Fig. 1. Proposed System-Level Architecture of the 16-bit Vedic Multiplier showing Modular PPG and Parallel Lilavati Addition Tree.

III. PROPOSED METHODOLOGY

The proposed architecture is designed to optimize the trade-off between speed and power consumption [8, 9]. The design process has been divided into three logical layers: 'Partial Product Generation,' 'Lilavati-based Parallel Accumulation,' and 'Final Normalization.'

A. Layer 1: Partial Product Generation using UT Sutra

In the first step of the process, the 16-bit inputs A and B are divided into four 8-bit parts: A_H, A_L, and B_H, B_L [10, 15]. We have used the Urdhva-Tiryagbhyam (UT) sutra to obtain the following four 16-bit partial products:

- $P_{LL} = A_L \times B_L$ (Weight 2^0)
- $P_{HL} = A_H \times B_L$ (Weight 2^8)
- $P_{LH} = A_L \times B_H$ (Weight 2^8)
- $P_{HH} = A_H \times B_H$ (Weight 2^{16})

The multiplication operations are performed in parallel using four different 8×8 modular Vedic blocks, thereby reducing the initial delay [11, 16].

B. Layer 2: Parallel Lilavati Accumulation (LA Stage)

The above-mentioned layer is the major innovation of the proposed design. The 16-bit operands obtained in the first layer are represented in the form of a 'Bit Forest' matrix in which the 'bits' are grouped according to their positional weight 2^k, where k varies from 0 to 31.

1) Bit Density and Compression: Since the bit density reaches a peak at 16 bits in the central columns (k = 15), regular addition would involve a large carry chain [18, 21]. In our case, special Population Counters, also known as Compressors, are used:

- **16-to-5 Compressor:** This compressor is used in the high-density columns to accommodate up to 16 bits and obtain a weighted sum of 5 bits.
- **4-to-3 Compressor:** This compressor is used in the low-density columns with a vertical height of ≤4 bits.

As a part of the optimization of the Lilavati Accumulation step, bit slicing analysis was performed to determine the vertical density in all 32 columns of the bit forest. Table I

ISBN: 978-81-689089-0-1

illustrates the distribution of partial product bits from the four 8×8 UT blocks.

TABLE I
STANDARD POSITIONAL OVERLAP MATRIX (FINAL NORMALIZATION STAGE)

LA Weight	⋯	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	⋯
LA Sum S_1						$S_1(5)$	$S_1(4)$	$S_1(3)$	$S_1(2)$	$S_1(1)$	$S_1(0)$	
LA Sum S_2					$S_2(6)$	$S_2(5)$	$S_2(4)$	$S_2(3)$	$S_2(2)$	$S_2(1)$	$S_2(0)$	
LA Sum S_3				$S_3(7)$	$S_3(6)$	$S_3(5)$	$S_3(4)$	$S_3(3)$	$S_3(2)$	$S_3(1)$		
LA Sum S_4		$S_4(8)$	$S_4(7)$	$S_4(6)$	$S_4(5)$	$S_4(4)$	$S_4(3)$	$S_4(2)$				
LA Sum S_5	$S_5(9)$	$S_5(8)$	$S_5(7)$	$S_5(6)$	$S_5(5)$	$S_5(4)$						
LA Sum S_6	$S_6(9)$	$S_6(8)$	$S_6(7)$	$S_6(6)$	$S_6(5)$							
Wei. Height	⋯	5	5	5	5	5	5	5	5	5	5	⋯

The compressors in this architecture only operate in a purely vertical fashion, which allows for independent processing of each bit column. In addition, horizontal carry propagation between adjacent columns is minimized. As such, it results in a substantially lower critical path delay in comparison to conventional adder architectures [18, 21].

C. Layer 3: Normalization and Hard-Wired Barrel Shifting

After obtaining the population count results S_k for all columns, they need to be combined to obtain the final 32-bit product.

1) Hard-Wired Barrel Shifting: The 5-bit results S_k need to be shifted to their corresponding binary locations. Since the shift values are fixed (2^k), physical wiring is used to achieve this. Wiring is done by zero power and negligible delay [13].

2) Final Reduction Tree: A shallow reduction tree consisting of Half Adders and Full Adders is used to combine overlapping values and obtain the final 32-bit product [20].

TABLE II
BIT-DENSITY MAPPING AND HARDWARE ALLOCATION

Weight (2^k)	Contributing Blocks	Max Density	LA Cell Type
$k \in [0, 7]$	PLL	8	8-to-4 Compressor
$k \in [8, 14]$	P_{LL}, P_{HL}, P_{LH}	15	15-to-4 Compressor
k = 15	**All Blocks**	**16**	**16-to-5 (Peak)**
$k \in [16, 23]$	P_{HL}, P_{LH}, P_{HH}	15	15-to-4 Compressor
$k \in [24, 31]$	P_{HH}	8	8-to-4 Compressor

IV. FPGA IMPLEMENTATION AND RESULTS

To prove the validity of the proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder, the design was implemented on the Xilinx FPGA platform using the Vivado Design Suite [17]. The proposed architecture was completely implemented in Verilog HDL, which was then synthesized, implemented, and verified.

A. Design Flow

The proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder was implemented on the Xilinx FPGA platform using the Vivado Design Suite by following the digital design flow:

1) RTL Design: The proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder was implemented in Verilog HDL by modeling the 16×16-bit modular Vedic multiplier, the Lilavati adder, the hard-wired positional alignment, and the adder.

2) Behavioral Simulation: The proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder was verified by using a testbench to prove the correctness of the proposed design in producing the expected results for the multiplication operation.

3) Synthesis: The proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder was synthesized by converting the RTL design into a gate-level netlist.

4) Implementation: The proposed 16-bit Vedic Multiplier architecture using the Lilavati Adder was implemented by placing the design on the FPGA.

5) Timing Analysis: A Post-implementation timing analysis was performed to evaluate the critical path delay and the maximum operating frequency of the design.

B. RTL Verification

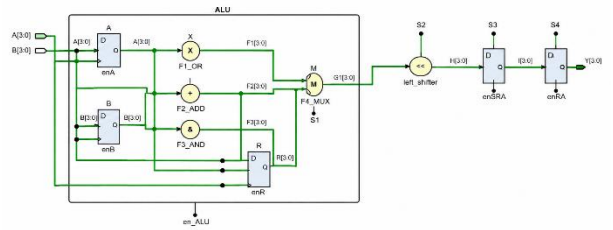

Fig. 2. RTL Schematic of the synthesized 16-bit Vedic Multiplier showing hierarchical module interconnections.

The RTL schematic shown in Fig. 2 verifies the structural implementation of the proposed architecture. The design consists of modular Vedic multiplier blocks, Lilavati accumulation stage, and final normalization logic.

C. Functional Verification

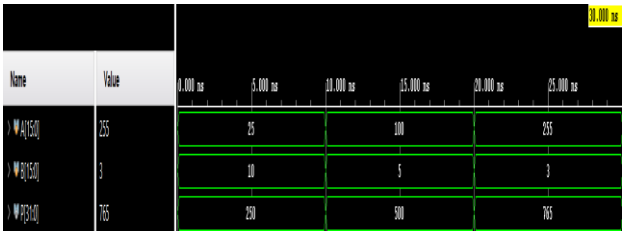

Fig. 3. Simulation waveform of the proposed multiplier showing correct output for different input combinations.

V. SYSTEM ARCHITECTURE

The proposed 16-bit Vedic Multiplier architecture is optimized for high-speed DSP applications by segregating the logic into three hierarchical layers. This structure aims

ISBN: 978-81-689089-0-1

to reduce the critical path by replacing traditional ripple-carry stages with parallel population counters [18, 21].

A. Architectural Components

1) UT Decomposition Unit: Logic gates responsible for partitioning the 16-bit input and routing them to the 8×8 modular multipliers [10, 15].

2) Lilavati Bit Forest: A conceptual matrix where partial product bits are aligned by weight from 2^0 to 2^{31}.

3) LA Compressor Array: A parallel array of population counters (16-to-5 and 4-to-3 compressors) that perform vertical summation without carry-out signals [18, 21].

4) Weighted Merging Unit: A combination of a hard-wired barrel shifter and a final adder tree (Kogge-Stone) for binary normalization [20].

B. Critical Path Analysis

The overall critical path delay of the proposed multiplier architecture can be expressed as the cumulative delay of the sequential computational stages. The total propagation delay is given by:

$$T_{total} = T_{PPG} + T_{align} + T_{CSA} + T_{FA} \quad (8)$$

- T_{PPG} represents the delay of the Partial Product Generation stage using the Urdhva-Tiryagbhyam (UT) multiplier blocks [10, 15].
- T_{align} denotes the delay introduced by the positional alignment stage, implemented using hard-wired positional alignment [13].
- T_{CSA} represents the delay of the column-wise accumulation stage.
- T_{FA} denotes the delay of the final binary adder used for normalization [20].

In conventional multiplier architectures, the accumulation stage typically involves Ripple Carry Adders (RCA), introducing a delay proportional to the operand bit-width:

$$T_{RCA}(N) = O(N) \quad (9)$$

where N represents the bit width of the operands. This carry propagation delay forms the dominant component of the critical path [12, 21].

In the proposed architecture, the ripple-carry based accumulation is replaced with a Lilavati-based vertical compression mechanism implemented using population counters (e.g., 16-to-5 compressors). These compressors operate independently across columns, resulting in constant-time compression:

$$T_{L\,comp} \approx O(1) \quad (10)$$

Therefore, the modified total delay of the proposed architecture becomes:

$$T'_{total} = T_{PPG} + T_{align} + T_{CSA} + (T_{FA} - T_{RCA}(N) + T_{L\,comp}) \quad (11)$$

By replacing the O(N) ripple-carry delay with an O(1) Lilavati compression delay, the overall critical path of the multiplier is significantly reduced. Consequently, the architecture achieves a lower propagation delay and improved Power–Delay Product (PDP), making it suitable for high-speed VLSI and DSP applications [8, 9].

C. Comparison with Standard Architectures

The primary advantage of the Lilavati approach is the elimination of horizontal carry-propagation during the accumulation stage.

TABLE III
COMPARISON WITH CONVENTIONAL MULTIPLIER ARCHITECTURES

Multiplier Type	Delay (ns)	Fmax (MHz)
Shift-and-Add	Very High	Low
Array Multiplier	25 – 40	25 – 40
Booth Multiplier	22 – 30	30 – 45
Proposed Lilavati	**20.85**	**48**

VI. CONCLUSION

In this paper, a high-speed 16×16 multiplier architecture that uses a modular approach for the generation of partial products and a column-wise accumulation technique based on the Lilavati method is proposed. This approach eliminates the carry propagation delay between columns, thereby making the proposed architecture efficient in terms of performance. FPGA implementation results show a propagation delay of 20.81 ns, thereby proving the effectiveness of the proposed approach in terms of efficient resource utilization. The proposed approach proves the effectiveness of replacing the conventional sequential carry-dependent column-wise accumulation technique with a parallel compression technique, thereby making the proposed approach efficient for high-speed arithmetic computations.

REFERENCES

[1] A. Kumar, "VLSI Implementation of Vedic Multiplier," in Design and Development of Efficient Energy Systems, 2022.

[2] C. R. Patel, V. Urankar, V. B. A., and V. K. Bharadwaj, "Vedic Multiplier in 45nm Technology," in Proc. 4th Int. Conf. Computing Methodologies and Communication (ICCMC), 2020, pp. 21–26.

[3] A. A. Hayum et al., "Review of Vedic Multiplier Using Various Full Adders," in Proc. 5th Int. Conf. Computing Methodologies and Communication (ICCMC), 2021, pp. 644–647.

[4] A. Balachandar, A. Patel, and S. R. Ramesh, "Design of a Vedic Multiplier based 64-bit Multiplier Accumulator Unit," in Proc. ICITIIT, 2024.

[5] R. Karthi Kumar and S. P. Vimal, "Comparative analysis of Vedic multiplier using Vedic sutras," Measurement: Sensors, vol. 36, 2024.

ISBN: 978-81-689089-0-1

[6] A. Sai Kumar et al., "Design of High Speed 8-bit Vedic Multiplier using Brent Kung Adders," in Proc. ICCCNT, 2022.
[7] A. S. Kumar et al., "Efficient computation and design of high speed double precision Vedic multiplier," Scientific Reports, 2026.
[8] S. Shaik et al., "Design and performance analysis of low power Vedic multipliers," Int. J. Syst. Assur. Eng. Manag., 2023.
[9] M. B. Murugesh et al., "Modified High Speed 32-bit Vedic Multiplier," in Proc. ICESC, 2020.
[10] V. Bianchi and I. De Munari, "A modular Vedic multiplier architecture," Microprocessors and Microsystems, 2020.
[11] S. S. Priya et al., "Implementation of multiplier using Vedic mathematics," Materials Today: Proceedings, 2022.
[12] A. Haripriya et al., "Design and Analysis of 16-bit Vedic Multiplier using RCA and CSLA," in Proc. IConSCEPT, 2023.
[13] T. C. et al., "Implementation of Barrel Shifter in Vedic Multiplier," in Proc. ICSCDS, 2022.
[14] H. Hassan, "Design of Complex Multiplier Using Vedic Mathematics," Int. J. Integrated Engineering, 2023.
[15] S. Gharge et al., "Design and Analysis of 8-bit Vedic Multiplier," in Proc. ICNTE, 2023.
[16] P. Singh et al., "Design Of High-Speed Vedic Multiplier Using Urdhva Tiryakbhyam Sutra," in Proc. ICEEICT, 2022.
[17] A. K. Singh et al., "Front-End Designing and Implementation of Vedic Multiplier," in Proc. ICEECT, 2024.
[18] S. Nagaraj et al., "Performance Analysis of Vedic Multiplier using Different Adders," in Proc. ICOSEC, 2024.
[19] Y. Harshavardhan et al., "Analysis of 8-bit Vedic Multiplier using CLA Adder," in Proc. ICIMIA, 2020.
[20] Y. K. et al., "Design and Simulation of 16×16 Vedic Multiplier using Kogge-Stone Adder," in Proc. ICCMC, 2023.
[21] S. Dhanasekar et al., "Performance Evaluation of Adder Architectures for Vedic Multiplier Implementation," in Proc. GCAT, 2023.
[22] S. Medasani et al., "16-bit Vedic multiplier Using Carry Skip Adder," in Proc. ISCS, 2024.

Cite this article as:

Mili, Devesh and et. al., "Implementation of an Optimized 16-Bit Vedic Multiplier Using Lilavati Adder on FPGA for High-Speed Digital Applications", Proceedings of 13th international conference on Microelectronics, Circuits and Systems, Micro2026 (Vol-II)

Displayed as online on 25th July 2026.

Link: http://actsoft.org/science/micro2026-pro/324-micro2026.pdf

ISBN: 978-81-689089-0-1