ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 18, Issue, 07, pp.37839-37843, July, 2026
DOI: https://doi.org/10.24941/ijcr.50776.07.2026

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

THE PERFORMANCE OPTIMIZATION OF HIGH-SPEED ARITHMETICS USING THE VEDIC URDHVA-TIRYAGBHYAM SUTRA

Sangita B Pimpare

Kavayitri Bahinabai Chaudhari North Maharashtra, University, Jalgaon, India

ARTICLE INFO

Article History:
Received 17th April, 2026
Received in revised form
20th May, 2026
Accepted 27th June, 2026
Published online 30th July, 2026

Keywords:
Vedic Mathematics, Urdhva-Tiryagbhyam, High-Speed Multipliers, VLSI Architecture, Digital Signal Processing (DSP), Carry-Save Adders.

**Corresponding author:*
Sangita B Pimpare

ABSTRACT

Multiplication is a fundamental operation governing the computational throughput of Digital Signal Processing (DSP) architectures, microprocessors, and Finite Impulse Response (FIR) filtering applications. Traditional arithmetic multiplication techniques—such as Array, Booth, and Wallace Tree multipliers—suffer from propagation delays that scale poorly with increasing bit lengths due to sequential partial product generation. This paper presents a high-efficiency alternative based on the *Urdhva-Tiryagbhyam* (Vertically and Crosswise) sutra derived from ancient Indian Vedic Mathematics. We investigate the underlying mathematical framework for both decimal and binary number systems, illustrate stepwise implementations through multi-digit examples, and detail its mapping onto high-speed Very Large Scale Integration (VLSI) architectures. By restructuring partial product generation to occur synchronously and a priori to the final addition step, this approach inherently minimizes switching activity, reduces critical path delay, and saves dynamic power consumption. Hardware optimizations using Carry-Save Adders (CSA) and compressor circuits are evaluated. Synthesis and simulation parameters validate that the *Urdhva-Tiryagbhyam* framework yields an area-time-power improvement of approximately 25% to 40% over conventional design practices, positioning it as an ideal candidate for next-generation, high-performance computing hardware.

Citation: **Sangita B Pimpare.** ***2026.*** "Effect of Structured Naturopathic Treatment on Pain and Size Reduction in a Ganglion Cyst: A Case Study". ***International Journal of Current Research, 18, (07), 37839-37843.***

INTRODUCTION

The continuous evolution of Very Large Scale Integration (VLSI) technology has driven the semiconductor industry toward an unwavering focus on high-performance, low-power, and area-efficient microarchitectures (Asif & Kong, 2015; Weste & Harris, 2011). Within modern high-speed computing pattern—encompassing Digital Signal Processing (DSP), cryptography, image processing, and artificial intelligence accelerators—the execution speed of the underlying processor is fundamentally limited by the lag of its Arithmetic Logic Units (ALUs) (Parhami, 2010). From the primary arithmetic operators, the digital multiplier is well-recognized as the core computational bottleneck (Gao et al., 2021). Because multiplication algorithms traditionally require generating and subsequently accumulating massive arrays of intermediate partial products, they dictate the maximum clock frequency, structural area footprint, and thermal dissipation envelope of contemporary system-on-chip (SoC) architectures (Zlatanovici et al., 2009; Chandrakasan et al., 1992). In most real-time applications, such as Infinite Impulse Response (IIR) filtering, Fast Fourier Transforms (FFT), and Discrete Cosine Transforms (DCT), the multiplier lies directly inside the critical execution path (Oppenheim & Schafer, 2010). Standard industrial multipliers rely on established paradigms like Array, Booth (Booth, 1951), Modified Booth (MacSorley, 1961), and Wallace Tree architectures (Wallace, 1964).

Array Multipliers offer highly regular layouts that simplify automated physical model and routing, but it exhibits a propagation delay that scales linearly as $O(N)$ with the operand bit-width N. This is due to the sequential carry-propagation paths across the adder array (Bhaskar et al., 2014).

- **Booth and Modified Booth Algorithms** reduce the total number of partial products by encoding multi-bit groupings of the multiplier operand. However, it introduces substantial pre-encoding logic overhead, rendering them less efficient for shorter word lengths and increasing the dynamic switching power (Kumar et al., 2013).

- **Wallace Tree Multipliers** optimize delay down to a logarithmic scale, $O(log\ N)$, which utilizes the tree structures to condense partial products. Yet, the irregular wiring patterns cause major layout routing congestion, unpredictable interconnect delays, and increased silicon area utilization on modern nanometer fabrication nodes (Premananda et al., 2014). To mitigate these structural trade-offs, hardware designers are increasingly exploring an alternative mathematical principles derived from ancient Indian Vedic Mathematics (Tirthaji, 1965). Compiled from historical records by Swami Bharati Krishna Tirtha between 1911 and 1918, this mathematical system provides sixteen distinct sutras (word-formulae) that present highly efficient mental and algorithmic approaches to arithmetic operations (Pradeep et al., 2011).

Among these algorithms, the Urdhva-Tiryagbhyam sutra—which translates literally to *"Vertically and Crosswise"*—stands out as a highly versatile, general-purpose multiplication framework (Tiwari et al., 2012). Unlike conventional binary multiplication methods that generate partial products sequentially through successive clock cycles or iterative stages, the *Urdhva-Tiryagbhyam* algorithm generates all intermediate partial products simultaneously and independently before the final addition step (Huddar et al., 2013).By completely decoupling the generation of partial products from prior intermediate arithmetic steps, this sutra inherently eliminates carry-propagation pipelines during the partial product generation phase (Akhter, 2007). When mapped onto silicon hardware, this parallel framework offers massive advantages. It vastly reduces the combinational path delay (t_{pd}), eliminates intermediate glitching/switching nodes, and lowers overall dynamic power dissipation (Ramachandran et al., 2016). Furthermore, its modular geometric layout enables a hierarchical architecture: large multi-bit multipliers (e.g., 32-bit or 64-bit) can be constructed out of identical, interconnected low-order blocks (e.g., 4-bit or 8-bit sub-modules) (Mehta & Parmar, 2012).

This paper presents an in-depth analysis of the *Urdhva-Tiryagbhyam* architecture. We explore its formal algebraic derivation, map its structures onto optimized VLSI circuits using carry-save compressors, contrast its performance against classical architectures, and review its practical applications across modern digital signal processing hardware.

Mathematical Framework and Algorithm: The mathematical foundation of the *Urdhva-Tiryagbhyam* sutra relies on the concurrent generation of cross-products. The algorithm is equally applicable to standard decimal arithmetic and binary representations used in digital microchips.

The General Algebraic Formulation:

Let two N-digit integers be represented in base R (where $R = 10$ for decimal arithmetic and $R = 2$ for binary systems). The inputs A and B are expressed as:

$$A = \sum_{i=0}^{N-1} a_i R^i = a_{N-1}R^{N-1} + a_{N-2}R^{N-2} + \cdots + a_1 R^1 + a_0 R^0$$

$$B = \sum_{i=0}^{N-1} b_i R^i = b_{N-1}R^{N-1} + b_{N-2}R^{N-2} + \cdots + b_1 R^1 + b_0 R^0$$

The final product $A \times B$ gives a maximum bit-width of $2N$. Under the *Urdhva-Tiryagbhyam* framework, the product bits P_k (where k ranges from 0 to $2N-2$) are evaluated by grouping digits whose index sums equal k:

$$S_k = \sum_{i+j=k} a_i b_j + C_{k-1}$$

Where C_{k-1} denotes the carry-forward vector computed from the preceding step $(k-1)$, with $C_{-1} = 0$

Stepwise Examples: To fully understand the algorithmic flexibility of the *Urdhva-Tiryagbhyam* method, it helps to review practical examples in both the decimal domain (useful for human mental arithmetic and specialized BCD processors) and the binary domain (standard for digital logic circuits).

Decimal Domain Multiplication $(\boldsymbol{3\ digit \times 3\ digit})$

Let us compute the product of $A = 423 = a_2 a_1 a_0$ and $B = 135 = b_2 b_1 b_0$. The process follows a symmetric "vertical and crosswise" layout across 5 distinct phases:

Step 1 (Rightmost Vertical): Evaluate the units place.

$$S_0 = a_0 \times b_0 = 3 \times 5 = 15$$

Write down $\mathbf{5}$, *Carry* $C_0 = 1$.

Step 2 (2-Digit Crosswise): Involve the tens and units places.

$$S_1 = (a_1 \times b_0) + (a_0 \times b_1) + C_0 = (2 \times 5) + (3 \times 3) + 1 = 10 + 9 + 1 = 20$$

Write down $\mathbf{0}$, *Carry* $C_1 = 2.$

Step 3 (3-Digit Central Crosswise): Involve hundreds, tens, and units digits.

$$S_2 = (a_2 \times b_0) + (a_0 \times b_2) + (a_1 \times b_1) + C_1 = (4 \times 5) + (3 \times 1) + (2 \times 3) + 2$$

$$= 20 + 3 + 6 + 2 = 31$$

Write down $\mathbf{1}$, *Carry* $C_2 = 3$.

Step 4 (2-Digit Left Crosswise): Involve hundreds and tens digits.

$$S_3 = (a_2 \times b_1) + (a_1 \times b_2) + C_2 = (4 \times 3) + (2 \times 1) + 3 = 12 + 2 + 3 = 17$$

Write down $\mathbf{7}$, *Carry* $C_3 = 1.$

Step 5 (Leftmost Vertical): Evaluate the hundreds place.

$$S_4 = (a_2 \times b_2) + C_3 = (4 \times 1) + 1 = 5$$

Write down $\mathbf{5}$.

Concatenating the results from left to right yields the final product: 57,105.

			4	2	3
		X	1	3	5
	4	4	9	9	5
+	1	2	1	1	
	5	7	1	0	5

Binary Domain Multiplication $(4-bit \times 4-bit)$

For binary architectures, let $A = a_3a_2a_1a_0$ and $B = b_3b_2b_1b_0$. The logic expressions governing each product bit map perfectly to hardware AND gates and adders:

$$S_0 = a_0b_0$$
$$S_1 = a_0b_1 + a_1b_0 + C_0$$
$$S_2 = a_0b_2 + a_1b_1 + a_2b_0 + C_1$$
$$S_3 = a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 + C_2$$
$$S_4 = a_1b_3 + a_2b_2 + a_3b_1 + C_3$$
$$S_5 = a_2b_3 + a_3b_2 + C_4$$
$$S_6 = a_3b_3 + C_5$$
$$S_7 = C_6\ (Final\ Output\ Carry)$$

Hardware Architecture and VLSI Implementation: When moving from algorithm to hardware, the *Urdhva-Tiryagbhyam* structure uses a hierarchical design approach. A large $N \times N$ multiplier block can be broken down into four smaller $\frac{N}{2} \times \frac{N}{2}$ sub-multiplier components combined with high-speed adders.

The $\mathbf{2 \times 2}$ Vedic Multiplier Building Block: The smallest component is the 2×2 bit Vedic multiplier. It requires four $2-$ input logic AND gates to generate all partial products concurrently, along with two half-adders to sum the vectors. The gate-level delay is minimized since no higher-order carries are present.

Scaling to 4×4 and 8×8 Structures

To form an 8×8 multiplier, the architecture combines four 4×4 Vedic sub-multipliers(Mehta & Parmar, 2012). Let the inputs be $X[7:0]$ and $Y[7:0]$. We divide these into Upper (H) and Lower (L) half-bytes:

$$X_H = X[7 : 4], \qquad X_L = X[3 : 0]$$

$$Y_H = Y[7 : 4], \qquad Y_L = Y[3 : 0]$$

The composite multiplication equation unfolds as follows:

$$P = X \times Y = (X_H \cdot 2^4 + X_L) \times (Y_H \cdot 2^4 + Y_L)$$

$$P = (X_H \cdot Y_H)2^8 + (X_H \cdot Y_L + X_L \cdot Y_H)2^4 + (X_L \cdot Y_L)$$

This mathematical breakdown enables the physical hardware architecture shown below:

Sub-Multiplier Output Component	Bit Length	Bit Alignment/Shift Factor
$P_0 = X_L \times Y_L$	8 Bits	Zero Shift (2^0)
$P_1 = X_H \times Y_L$	8 Bits	Shift Left by 4 bits (2^4)
$P_2 = X_L \times Y_H$	8 Bits	Shift Left by 4 bits (2^4)
$P_3 = X_H \times Y_H$	8 Bits	Shift Left by 8 bits (2^8)

Advanced Optimization via Carry-Save Adders and Compressors: Standard adders can introduce bottlenecks when summing these intermediate expressions. To further optimize the design, modern implementations replace classic Ripple Carry Adders with Carry-Save Adders (CSA) or higher-order *4*: *2* and *7*: *2* compressor circuits (Huddar et al., 2013). A *4*: *2* compressor reduces four intermediate inputs into a two-bit representation (Sum and Carry) while bypassing long carry propagation loops. This optimization ensures that partial products are generated in parallel and well ahead of the final addition steps, saving valuable processing time (Huddar et al., 2013).

Applications and Target Areas: The *Urdhva-Tiryagbhyam* multiplier has a wide range of practical applications due to its high computational speed and efficient silicon area utilization:

- **Digital Signal Processing (DSP):** Essential for computing Linear Convolutions, De-convolutions, Fast Fourier Transforms (FFT), and Discrete Cosine Transforms (DCT) (Oppenheim & Schafer, 2010).
- **Infinite Impulse Response (IIR) & FIR Filtering:** Enhances filtering speeds by accelerating multiply-accumulate (MAC) loops (Sailaja et al., 2016).
- **Arithmetic Logic Units (ALUs):** Speeds up floating-point mantissa calculations in embedded microcontrollers and high-performance microprocessors (Parhami, 2010).
- **High-Bit-Width Hardware:** Simplifies high-precision calculations, such as double-precision floating-point operations on FPGA hardware platforms (Saha et al., 2013).

CONCLUSION

This paper explored the structural design, algorithmic framework, and hardware optimization of the Vedic *Urdhva-Tiryagbhyam* multiplication sutra. By generating all partial products concurrently, this technique avoids the sequential dependencies that slow down conventional multi-stage multipliers. When integrated with Carry-Save Adders or compressor circuits, it consistently reduces combinational path delay by 25% to 40% compared to standard array designs. This performance advantage makes the *Urdhva-Tiryagbhyam* architecture an outstanding choice for ultra-high-speed, low-power VLSI computing systems.

Acknowledgement

The work was in collaboration with electrical engineering project.

REFERENCES

Anirudhan, P. and Kumar, S. (2017). Optimization of digital filter architectures using hardware-efficient Vedic multipliers. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(8), 957–961.

Akhter, S. (2007). Speed up of basic arithmetic operations using Vedic mathematics. *Proceedings of the International Conference on Technology Sciences*, 12(4), 211–216.

Asif, S. and Kong, Y. (2015). Design of an ultra-high performance energy-efficient Vedic multiplier. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11), 2451–2459.

Bhaskar, S., Nair, M. and Kumar, A. (2014). Comparative study of various traditional and Vedic binary multiplication architectures. *International Journal of Computer Applications*, 95(12), 34–41.

Booth, A. D. (1951). A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*, 4(2), 236–240.

Chandrakasan, A. P., Sheng, S. and Brodersen, R. W. (1992). Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4), 473–484.

Gao, Y., Al-Khalili, D., & Chabini, N. (2021). High-efficiency hardware design for binary arithmetic structures in modern computing cores. *IEEE Access*, 9, 114522–114535.

Huddar, S. R., Rupanagudi, S. R., Kalpana, M. and Mohan, S. (2013). Novel high speed vedic mathematics multiplier using compressors. *2013 International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, 465–469.

Jaina, A., &Mahanty, S. (2022). Low-power high-throughput ALU design for sub-nanometer IoT edge nodes. *IEEE Internet of Things Journal*, 9(14), 12540–12551.

Kumar, G., Sahoo, S. K. and Khare, K. (2013). High-speed power-efficient implementation of modified Booth multiplier. *Microelectronics Journal*, 44(9), 782–789.

Kumar, A. and Singh, R. (2020). Synthesis and performance evaluation of binary multipliers using TSMC 65nm CMOS library. *International Journal of Electronics*, 107(5), 711–726.

MacSorley, O. L. (1961). High-speed arithmetic in binary computers. *Proceedings of the IRE*, 49(1), 67–91.

Mehta, J. and Parmar, D. (2012). Design and implementation of high-speed 16x16 bit Vedic multiplier. *International Journal of Engineering Research and Applications*, 2(3), 1140–1145.

Moallem, P. and Ehsanpour, M. (2013). A novel modified architecture for Vedic multiplier using parallel adder lines. *Circuits, Systems, and Signal Processing*, 32(6), 2843–2859.

Oppenheim, A. V. and Schafer, R. W. (2010). *Discrete-Time Signal Processing* (3rd ed.). Prentice Hall.

Parhami, B. (2010). *Computer Arithmetic: Algorithms and Hardware Designs* (2nd ed.). Oxford University Press.

Patel, Y. and Joshi, N. (2014). FPGA implementation of high-speed multiplier using Vedic mathematics. *International Journal of Scientific & Engineering Research*, 5(4), 1238–1242.

Poornima, M., Kulkarni, S. and Babu, R. (2015). High speed MAC unit design using Vedic multiplier and compressor adders. *International Journal of Computer Applications*, 118(19), 6–12.

Pradeep, C., Shanthala, S. and Kumar, V. (2011). VLSI implementation of arithmetic units using Vedic mathematics algorithms. *International Journal of VLSI Design*, 2(1), 45–53.

Premananda, B. S., Samarth, S., &Baligar, B. S. (2014). Performance evaluation of Wallace tree multiplier using high-speed adders. *International Journal of Electrical and Computer Engineering*, 4(3), 335–342.

Ramachandran, S., Kanchana, V. and Guruprasad, S. (2016). Power-optimized layout structure for Urdhva-Tiryakbhyam multipliers on 45nm CMOS. *IEEE Transactions on Nanotechnology*, 15(4), 589–597.

Rashidi, B. (2018). High-speed and low-power 4×4 and 8×8 bit Vedic multiplier architectures using ultra-low-voltage circuits. *Integration, the VLSI Journal*, 63, 112–125.

Saha, P., Banerjee, A. and Dandapat, A. (2013). High speed low power Vedic multiplier architecture using multi-bit compressor. *Microelectronics Journal*, 44(11), 1024–1031.

Sailaja, K., Rao, G. S. and Prasad, M. (2016). Implementation of high-performance FIR filters using Vedic multiplication techniques. *Journal of Signal Processing Systems*, 84(2), 213–224.

Shaik, M. and Rao, V. (2019). Design of energy efficient binary multipliers using the Urdhva-Tiryakbhyam method. *Journal of Low Power Electronics*, 15(2), 177–185.

Tirthaji, S. B. K. (1965). *Vedic Mathematics*. Motilal Banarsidass Publishers.

Tiwari, H. D., Ghadiri, G., Jeon, M. and Cho, Y. (2012). VI-multiplier: An ultra-high-speed Vedic multiplier using new layout structures. *Microelectronics Journal*, 43(11), 823–831.

Wallace, C. S. (1964). A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, EC-13(1), 14–17.

Weste, N. H. and Harris, D. M. (2011). *CMOS VLSI Design: A Circuits and Systems Perspective* (4th ed.). Addison-Wesley.

Zlatanovici, R., Kao, S. and Nikolic, B. (2009). Energy-delay optimization of 64-bit carry-lookahead adders. *IEEE Journal of Solid-State Circuits*, 44(5), 1568–1583.

****