INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

Design and Analysis of Nikhilam based Vedic Architecture for High Speed Arithmetic Operations on FPGA

Padum Kant Mishra[1], Ekta Gupta[2], Aniket Kumar[3]*

1 Swami Vivekanand Subharti University, Meerut, U.P., India
2 MIET, Meerut, U.P., India
3 Shobhit Institute of Engineering and Technology (Deemed to be University), Meerut, U.P., India

OPEN ACCESS

Received: 08/07/2025

Accepted: 21/07/2025

Published: 09/08/2025

Citation: Mishra PK, Gupta E, Kumar A (2025) Design and Analysis of Nikhilam based Vedic Architecture for High Speed Arithmetic Operations on FPGA. Indian Journal of Science and Technology 18(30): 2453-2459. https://doi.org/10.17485/IJST/v18i30.1208

* **Corresponding author**.

ani.vlsi2012@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: With rising computational demands in signal processing and machine learning, achieving high-throughput and low-latency multiplication on FPGA platforms is critical. This work targets the design, implementation, and synthesis results of 2-bit, 2^2-bit, 2^3-bit, 2^4-bit, and 2^5-bit binary multipliers using the Nikhilam Sutra of Early Indian Vedic Mathematics are explored in this paper. **Methods:** This study implements 2-bit, 2^2-bit, 2^3-bit, 2^4-bit, and 2^5-bit binary multipliers using VHDL Coding with reusable components such as AND gates, 2:2, 3:2 compressors and sub-multipliers. **Findings:** The coded design is implemented using Xilinx ISE 14.7 targeting spartan-3E FPGA, and afterwards performance matrices including utilization of RAM space, delay, LUTs & Level of Logic are compared across bit-width 2, 2^2, 2^3, 2^4 and 2^5. **Novelty:** By extending the design to 32 bits and systematically comparing resource tradeoffs, this study provides comprehensive insights into Nikhilam-sutra results of simulation and synthesis approve that the proposed Nikhilam-based multipliers are precise, scalable, and offer a balanced trade-off between speed and hardware complications, making them suitable for FPGA-based performance-critical applications.

Keywords: Adder; Multiplier; FPGA; Compressors; Response time; Nikhilam Sutra

1 Introduction

Natural signals such as speech, environmental sounds, and physical signals are inherently analog in nature. To control the advantages of modern digital processing, including improved noise protection, scalability, and computational accuracy, such signals are generally converted into digital form. This digitization is fundamental to a broad range of applications [(1)].

Among the fundamental operations in embedded computing and signal processing is **multiplication**, that is often the performance bottleneck due to its computational intensity and hardware cost. Consequently, optimizing multipliers has become a focal point of research aimed at reducing delay, power consumption, memory usage, and

hardware complexity[2]. Traditional multiplication approaches such as array and Booth multipliers provide functional accuracy but suffer from limitations in speed, area, or memory when scaled to higher bit-widths[3].

To beat these restraints, researchers have turned to ancient Vedic mathematics. These methods are demonstrated as propagation delay and gate-level complexity[4],[5]. Current studies have extended Vedic multiplication methods to FPGA platforms using hardware description languages (HDLs) like VHDL, Verilog and System VHDL, showing promising results for bit-widths up to 2^5 bits[6].

In spite of these developments, most prevailing implementations either focus narrowly on speed or neglect critical metrics such as **RAM usage, Look-Up Table (LUT) count**, and **logic level depth**—factors that significantly impact FPGA resource utilization and scalability in real-world applications[7].

In light of these constraints, this study presents a comprehensive design and evaluation of Vedic multipliers based on the Nikhilam algorithm across various bit-widths. The designs are implemented in VHDL and synthesized on FPGA to assess performance matrices including **RAM utilization, response time, LUT count, and logic level depth**.

1.1 Related Work

Urdhava Triyakbhyam Sutra

Urdhva Tiryakbhyam (UT), a multiplication technique from ancient Vedic mathematics. It has general-purpose multiplication formula applicable to all types of multiplication. UT, which means "vertically and crosswise," significantly reduces computation time[1]. The general method of multiplication using this technique is illustrated below[2].

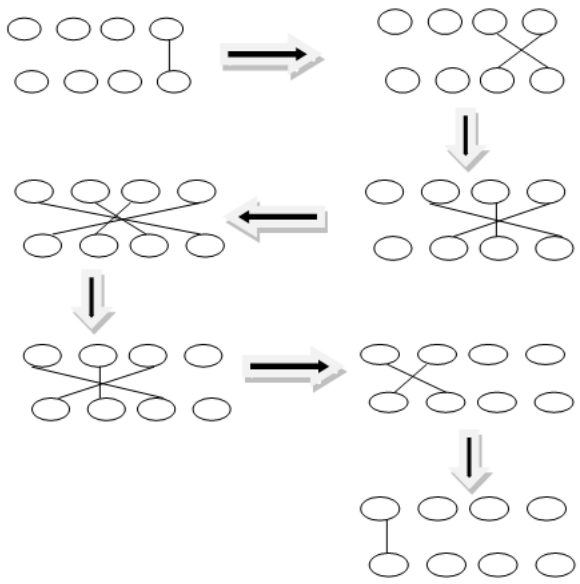

Fig 1. Multiplication process in Four Bit UT Multiplier

In Figure 1, the Urdhva Tiryakbhyam (UT) technique is employed to multiply a four-bit number using vertical and crosswise multiplication, progressing sequentially. A similar bit flow pattern is followed for higher-order multipliers, as above figure[3].

1.2 Nikhilam Sutra

The common approach of multiplication is both time-consuming and space-intensive, involving multiple steps. In contrast, the Vedic approach significantly trim down the number of steps and space required. Many problems can be solved in various ways, often mentally, due to the simplicity of the techniques. One such method is Nikhilam sutra, which allows problems with specific patterns to be solved in just one or two steps. However, a solid understanding of the base system—an essential concept in both Vedic and modern mathematics—is critical for using this approach effectively[8],[9],[10],[11].

In Figure 2, the subsequent moves that are followed while multiplying binary numbers is illustrated. It involves computing the deviation of two numbers from a base. These deviations are applied to split the answer into a Left-Hand Side (LHS) and

Right-Hand Side (RHS). The RHS is the product of deviations and contains as many digits as there are zeros in the base. The LHS is the sum of one number and the deviation of the other. If the RHS has fewer digits, leading zeros are added; if it exceeds, the extra digits are carried over to the LHS. Finally, the slash separating LHS and RHS is removed to obtain the final product [1].

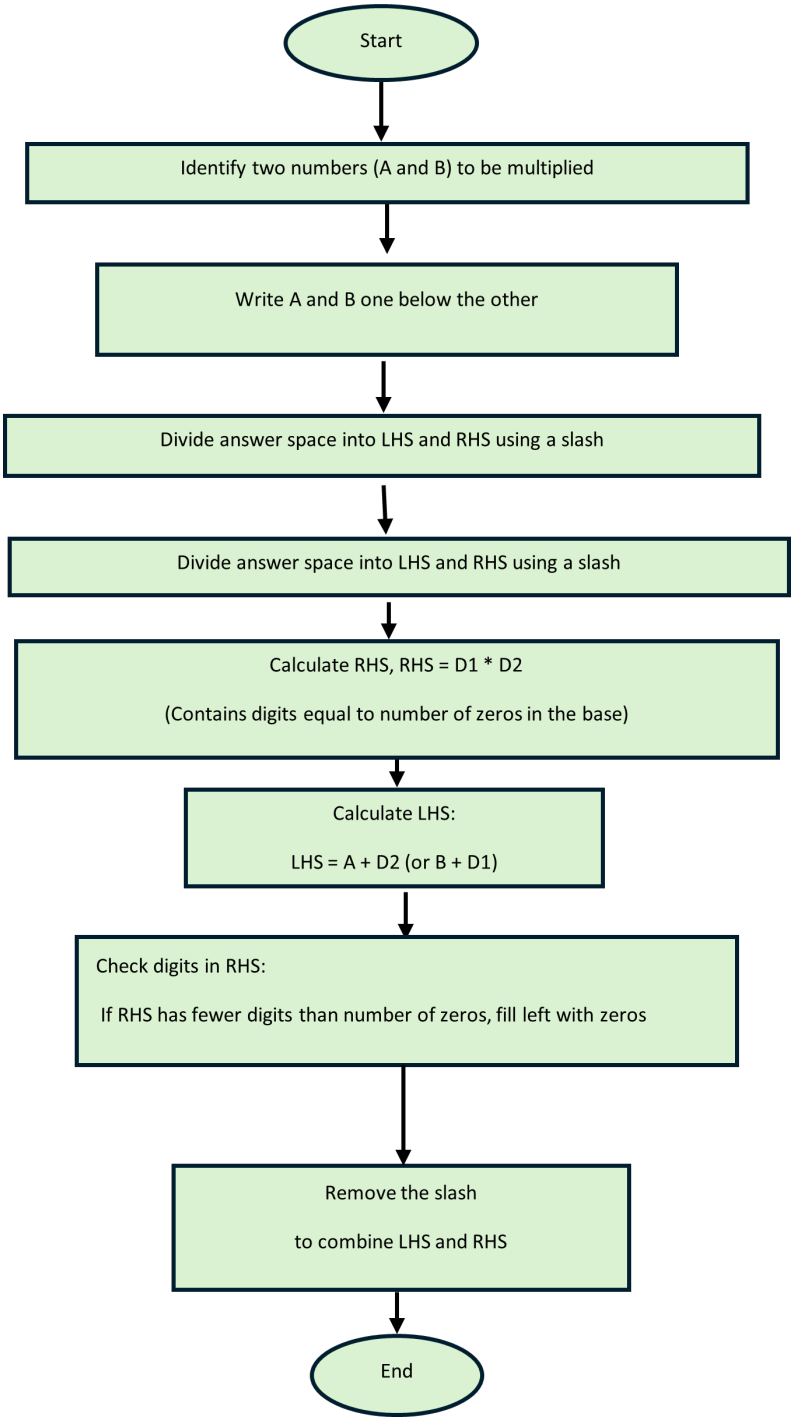

Fig 2. Process Flow of Nikhilam-Based Multiplication

2 Methodology

The Nikhilam code for bit length 2, 2^2, 2^3, 2^4 and 2^5 are written using VHDL and simulated using ModelSim-Intel FPGA Starter Edition 10.5d and on Xilinx 14.4 too, targeting the XC3SD1800A device from the Spartan-3A DSP family, with a CS484 package and a speed grade of -5.

3 Result and Discussion

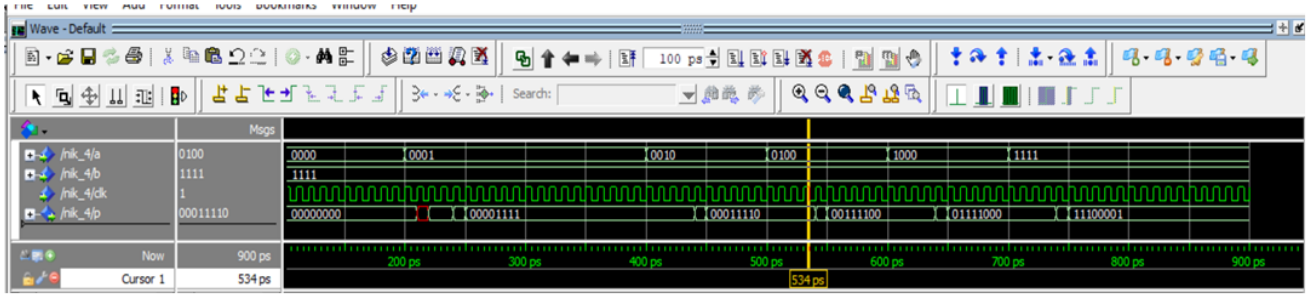

Fig 3. Simulation Result 4-bit Nikhilam Multiplier

In the waveform shown above, when value of 'a' is "0000" and 'b' is "1111", 8-bit result 'p' is "00000000", similarly when a= "1111" and b= "1111", result p is "11100001".

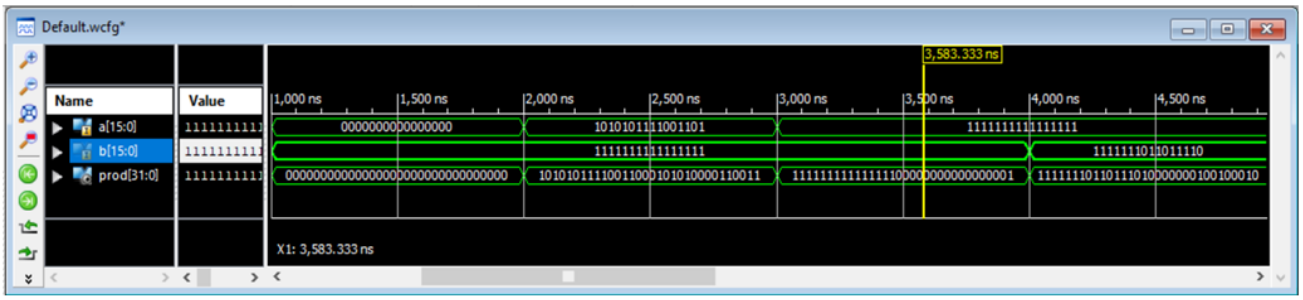

Fig 4. Simulation Result 16-bit Nikhilam Multiplier

As illustrated in waveform shown in Figure 4, when value of 'a' is 0000_{16} and 'b' is 0000_{16}, 32-bit result 'p' is 00000000_{16}, similarly when a= "$FFFF_{16}$" and b= "$FFFF_{16}$", result p is "FFFE0001".

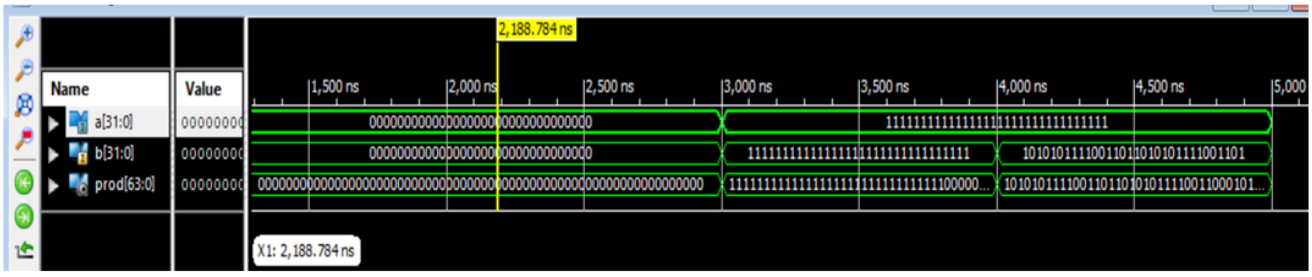

Fig 5. Simulation Result 32-bit Nikhilam Multiplier

As illustrated in waveform shown in Figure 5, when value of 'a' is 00000000_{16} and 'b' is 00000000_{16}, 32-bit result 'p' is 0000000000000000_{16}, similarly when a= "$FFFFFFFF_{16}$" and b= "$FFFFFFFF_{16}$", result p is "FFFFFFFE00000000".

thirtytwo_bit_multiplier Project Status (07/02/2025 - 11:06:32)			
Project File:	ab.xise	**Parser Errors:**	No Errors
Module Name:	thirtytwo_bit_multiplier	**Implementation State:**	Placed and Routed
Target Device:	xc3sd1800a-5cs484	**•Errors:**	No Errors
Product Version:	ISE 14.7	**•Warnings:**	282 Warnings (282 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	2,368	33,280	7%	
Number of occupied Slices	1,281	16,640	7%	
Number of Slices containing only related logic	1,281	1,281	100%	
Number of Slices containing unrelated logic	0	1,281	0%	
Total Number of 4 input LUTs	2,368	33,280	7%	
Number of bonded IOBs	128	309	41%	
Average Fanout of Non-Clock Nets	3.37			

Fig 6. Implementation Report of 32-bit Nikhilam Multiplier

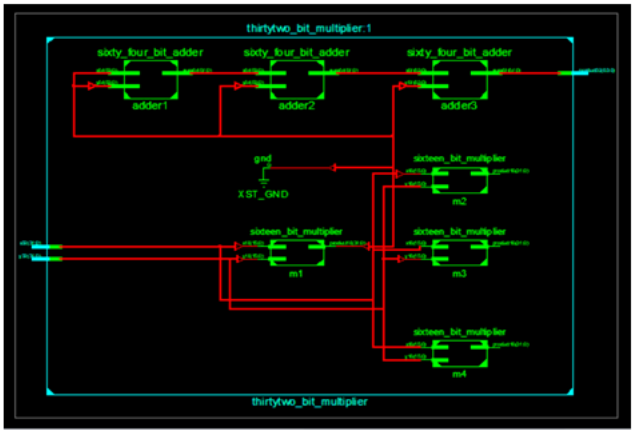

Fig 7. RTL Schematic of 32-bit Nikhilam Multiplier

As illustrated in Figures 6 and 7, the 32-bit Nikhilam multipliers utilize four 16-bit multiplier blocks and three 64-bit adder blocks, resulting in increased computation time and higher RAM usage.

The proposed architecture has been simulated using ModelSim Altera Edition 6.6d, where functional verification and waveform analysis have been performed to ensure correct behavioral operation. The hardware implementation has been completed using Xilinx ISE 14.4, targeting the Spartan-3A DSP FPGA family, specifically the XC3SD1800A device with a CS484 package and a -5-speed grade, which has provided enhanced performance characteristics.

The synthesis process has generated a detailed report that has provided key hardware metrics, including:

- Combinational delay, which has indicated the maximum propagation time through the logic.
- Levels of logic, which have represented the depth of the logic network.

- LUT count, which has reflected the number of functions implemented.
- Flip-flop usage, which has indicated the number of sequential elements used.
- Memory block usage, if any, has been taped for storage requirements.

Table 1. Comparative Analysis for various bit-length

Parameter	2-bit	4-bit	8-bit	16-bit	32-bit
Delay in ns	6.745	14.578	22.97	31.721	70.745
Logic Levels	3	11	21	27	66
LUTs Used	4	43	192	547	1357
Memory (KB)	4499672	4501016	4506136	4692224	4696720

These parameters were explored to conduct a relative study of multipliers across various operand bit-widths (such as 4-bit, 8-bit, 16-bit and 32-bit). This analysis has helped in evaluating trade-offs between area, speed, and power, thereby guiding the selection of optimal architectures for high-performance arithmetic units in FPGA-based digital systems.

To validate the proposed multiplier architecture, results were benchmarked against existing literature:

- A study by Banerjee et al.[12] reported that their 8-bit Vedic design consumed 233 LUTs and had a delay of 26.1 ns, compared to our **192 LUTs and 22.97 ns delay**, indicating better area and timing optimization
- Maheswari and Prabha[13] implemented a 16-bit multiplier using reversible logic, which offered power savings but resulted in a delay of 38.2 ns. Our design achieves **a faster 31.721 ns delay** at the cost of higher LUT usage, favoring speed over minimal area.
- However, in Dey and Chakraborty[14], a hierarchical 16-bit Vedic multiplier showed optimized memory usage (4.55 MB) but did not present logic level detail. Our results align closely in memory (4.69 MB), **but also quantify logic depth**, providing a more detailed hardware profile.
- LUTs used in 32-bit multiplier is 1357[15], which is less than the implemented architecture by Nishant G., making it **area-efficient in logic utilization.**

4 Conclusion

This finding magnificently demonstrates the design and FPGA implementation of Nikhilam-based Vedic multipliers across multiple bit-widths (2 to 32 bits). The results obtained from implementation, simulation, and comparison suggest that delay increases significantly across the multiplier size. Each subsequent bit size approximately adds nearly 8 ns which shows non-linear behavior due to growing circuit complexity as verified in **Table 1**. Memory usage increases progressively with bit size. The number of LUTs increases exponentially with subsequent bit size which reflects deeper circuit hierarchy, wider bid-widths and more functional blocks.

The simulation and synthesis results confirm that the proposed Nikhilam-based multipliers are functionally accurate and efficiently scalable. When benchmarked with related work, the architecture provides favorable trade-offs between speed and hardware complexity. This validates the viability of Nikhilam-based Vedic multipliers for use in performance-critical digital arithmetic units, especially in FPGA-based embedded systems where timing and resource usage are pivotal.

The novelty lies in extending and synthesizing Nikhilam Sutra-based designs to higher bit-widths (up to 32 bits) using reusable modular components, offering a balance between speed and area on FPGA.

Future prospects include optimizing logic depth, integrating pipelining for throughput improvement, and adapting the design for low-power applications.

Recommendations: This architecture is suitable for real-time, performance-critical applications in signal processing and machine learning. Further work can involve extending it to signed operations and ASIC implementation for wider deployment.

5 Declarations

Ethics Approval and Consent to Participate: Not applicable.

Consent for Publication: This manuscript does not contain any data related to individual persons.

Funding: No funding was received for this work.

Availability of Data and Materials: The VHDL code for the multipliers discussed in this study has been simulated and implemented using Xilinx tools.

Acknowledgements: All authors and co-authors contributed to simulation, coding, and result generation.

References

1) Mishra PK, Gupta E, Kumar A. Comparative Research on Power and Junction Temperature in 2, 4, 8 bit Multiplier Using Nikhilam Sutra. In: 2024 ICRASET, B G Nagara, Mandya, India. 2024;p. 1–4. https://doi.org/10.1109/ICRASET63057.2024.10895106.
2) Kumar A, Agarwal RP. Performance Evaluation and Synthesis of FIR Filters Using Various Multipliers Algorithms. Singapore. Springer. 2021. https://doi.org/10.1007/978-981-16-0749-3_45.
3) Kumar A, Agarwal RP. Design and Implementation of Efficient FIR Low Pass Filters Based on Vedic and Traditional Digital Multiplier Algorithm. Singapore. Springer. 2021. https://doi.org/10.1007/978-981-16-0942-8_25.
4) Patil A, Kapare S, Shinde G, Tekade A, Andhare M, Kumbar V. Create a 32-bit Vedic Multiplier and Compare it Against Other Multipliers Using A Carry Look-Ahead Adder. In: INCET, Belgaum, India. 2023;p. 1–4. https://doi.org/10.1109/INCET57972.2023.10170076.
5) Sharma D, Rai A, Debbarma S, Prakash O, Ojha MK, Nath V. Design and Optimization of 4-Bit Array Multiplier with Adiabatic Logic Using 65nm CMOS Technologies. *IETE Journal of Research*. 2023. https://doi.org/10.1080/03772063.2023.2204857.
6) Kumar A, Gupta E, Agarwal RP, Jain RK. Comparative Research for Managing Delay in Signal Processing Via Multipliers. In: Proc. of ICPIECES-2018, DTU, India. IEEE. 2018;p. 1549–1554. https://doi.org/10.1109/ICPEICES.2018.8897312.
7) Leela SN, Manisha B, Bharath P, Praneeth E. Design of Wallace tree multiplier circuit using high performance and low power full adder. *E3S Web of Conferences*. 2023;391:01025. https://doi.org/10.1051/e3sconf/202339101025.
8) Sharma V, Kumar A. Design, Implementation & Performance of Vedic Multiplier for Different Bit Lengths. *IJIRCCE*. 2017;5(4):7912–7919. Available from: https://www.researchgate.net/publication/316876747_Design_Implementation_Performance_of_Vedic_Multiplier_for_Different_Bit_Lengths.
9) Kumar A, Vishikha. Comparative Analysis of Vedic & Array Multiplier. *International Journal of Electronics and Communication Engineering and Technology*. 2017;8(3):17–27. Available from: https://oaji.net/articles/2017/2000-1528449826.pdf.
10) Kumar A, Agarwal RP. Complex Multiplier: Implementation using Efficient Algorithms for Signal Processing Application. *International Journal of Recent Technology and Engineering*. 2019;8:1235–1239. https://doi.org/10.35940/ijrte.C5147.118419.
11) Kumar A, Agarwal RP. Simulation & Implementation of Efficient Multiplier Circuits. In: Proceedings of ICRMR-2018, AIT Conference Centre, Thailand. 2018;p. 1–6. Available from: https://www.researchgate.net/publication/329799537_Simulation_and_Implementation_of_Efficient_Binary_Multiplier_Circuits.
12) Banerjee S. Comparative analysis of Vedic and conventional multipliers. *IET Circuits, Devices & Systems*. 2021;15(6):525–532. https://doi.org/10.1049/cds2.12041.
13) Maheswari M, Prabha S. Optimized Vedic Multiplier using Reversible Logic. *Microprocessors and Microsystems*. 2020;71.
14) Dey D, Chakraborty M. Efficient FPGA Implementation of Vedic Multiplier. In: IEEE INDICON. 2018. https://doi.org/10.1109/INFOP.2015.7489448.
15) Deshpande NG, Mahajan R. Ancient Indian Vedic Mathematics based 32-Bit Multiplier Design for High Speed and Low Power Processors. *International Journal of Computer Applications*. 2014;95(24). https://doi.org/10.5120/16742-6956.