Implementation and Performance Analysis of Vedic Multipliers on Nanoscale Power-Efficient FPGA Architectures

Sugandha Agarwal[1], Aniket Kumar[2]
[1,2]*School of Electronics, Electrical & Mechanical Engineering Shobhit Institute of Engineering and Technology (Deemed to be University), Meerut, Uttar Pradesh, India*

Abstract— **Multiplication is regarded as a fundamental arithmetic operation by which the speed, area, and energy efficiency of digital systems are directly affected. In this paper, a comparative implementation and performance analysis of five multiplier architectures that are Array, Booth, Dadda, Vedic, and Wallace is presented using VHDL and FPGA synthesis. These architectures are evaluated for operand widths of 2, 4, 8, and 16 bits across four Xilinx FPGA families: Artix-7, Kintex-7, Spartan-3E, and Virtex-7. Propagation delay, LUT utilization, and logical depth are devoted as the principal performing indicators, allowing both timing and hardware-resource requirements to be compared. It is shown by the results that the most beneficial timing performance for medium- and large-sized operands is provided by Dadda multiplication, whereas competitive resource utilization, particularly in smaller designs, is achieved by Vedic multiplication. Elevated delay and greater resource requirements are constantly exhibited by the Spartan-3E device than by the newer 7-series devices. It is therefore demonstrated that multiplier selection should be determined by the requirements of the intended application, with Dadda architectures being favored for timing-critical implementations and Vedic architectures being considered attractive for area-constrained designs.**

Index Terms— **FPGA; Vedic multiplier; Dadda multiplier; Booth multiplier; Wallace multiplier; Array multiplier; propagation delay; LUT utilization; digital arithmetic**

I. INTRODUCTION

Digital multipliers are fundamental building blocks in modern computational systems because processors and signal-processing hardware frequently execute multiplication. System speed and hardware efficiency in various applications such as image processing, machine-learning accelerators, cryptographic hardware, and scientific computing are directly influenced by multiplier performance. When multiplication lies on the critical path, an important element of attainable operating frequency and throughput is established by propagation delay.

In multiplier design, a trade-off among delay, resource utilization, and power consumption is involved. Real-time applications such as DSP, image compression, filtering, and encoding require efficient multiplication. Higher operating frequencies and improved throughput can be supported by lower propagation delay.

The hardware cost in FPGA is directly tied to logic cells, LUTs, slices, or gates consumed by the design. Resource-constrained and cost-sensitive applications are advantaged by compact multipliers. With the growing emphasis on portable and battery-operated devices, energy-efficient designs are increasingly regarded as important. Longer battery life, reduced thermal issues, and sustainable computing are contributed to by low-power multipliers. Numerous multiplier architectures have been developed to optimize different combinations of speed and resource utilization. Vedic multipliers, including architectures based on Urdhva-Tiryagbhyam and Nikhilam principles, are investigated as alternatives to conventional multiplier structures by exploiting parallel partial-product generation. However, the choice of multiplier architecture is not universal and must be aligned with the target application. The

effectiveness of a multiplier is varied with operand size, making it essential that evaluations of multiple architectures are implemented under uniform conditions and across different word lengths.

II. MULTIPLIER ALGORITHMS

The development of various algorithms like Vedic, Array, Wallace, Dadda, Booth, and Sequential, which offer different approaches to binary multiplication, is driven heavily by the reliance on multipliers in the design of digital filters.

A. Vedic Algorithm

The ancient Vedic sutra "Urdhva-Tiryagbhyam" ("vertical and crosswise") is employed by this design, an algorithm that calculates the product of two numbers through a series of vertical and crosswise multiplication steps[2].

B. Array Algorithm

Based on a sequential process of addition and shifting, partial products are generated by this algorithm. Rows are then arranged and finally summed column-wise to obtain the product terms [3].

C. Wallace Algorithm

To optimize the multiplication of large numbers, a tree-based architecture for partial product accumulation is employed by this algorithm. The critical path delay is minimized by implementing full adders and half adders as 3:2 and 2:2 compressors, respectively [4].

D. Dadda Algorithm

Partial products are also arranged in a tree structure by this algorithm.

The formula to $d_{j+1} = \text{floor}(1.5d_j)$, governs the reduction process, where d_j represents the maximum height of a column in the partial product matrix [5].

E. Booth Algorithm

In this alogorithm the number of partial products is reduced through an encoding technique. By examining the multiplier bits from MSB to LSB in groups, it is determined whether to add, subtract, or simply shift the multiplicand to obtain the result efficiently [6-8].

III. METHODOLOGY

All five multiplier algorithms were coded and simulated using VHDL on ModelSim and then the design was synthesized targeting four different Xilinx FPGA families using different technologies, i.e. Artix7, Kintex, Spartan3E, and Virtex7. A comparative evaluation for response time has been made for Array, Booth, Dadda, Vedic, and Wallace Algorithm post implementation.

IV. RESULT AND ANALYSIS

The evaluation of the performance of each multiplier is conducted from multiple angles in the synthesized results. The visualization of the most important trends is carried out in the comparative graphs for concise analysis.

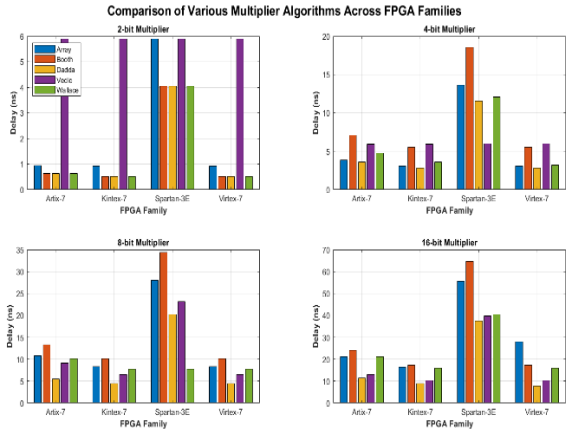

Fig. 1 Comparative Graph for delay

The comparative analysis obtained through MATLAB R2024b coding using simulation data, shows that significant increases in propagation delay are observed with multiplier bit size for all multiplier architectures and FPGA families. Among the FPGA platforms, lower delays are generally exhibited by Kintex-7 and Virtex-7, while the highest delay is shown by Spartan-3E, particularly for 16-bit multipliers. This indicates the impact of FPGA architecture and available high-speed logic resources on multiplier performance. The increase in delay with operand size is mainly attributed to the increased number of partial products and logic levels required for multiplication.

Among the multiplier architectures, the lowest delay is generally provided by Dadda for Artix-7 and Kintex-7, especially at higher bit sizes. Competitive performance is also demonstrated by Wallace, particularly for larger multipliers on Spartan-3E. Comparatively higher delays are exhibited by Booth at larger bit widths, while moderate performance is provided by Vedic. Overall, the results imply that the most favorable delay performance is offered by Dadda on Kintex-7 among the combinations evaluated, making it a promising choice for high-speed multiplier implementation.

TABLE I. Comparative RAM Analysis of Multiplier on Artix 7 , Kintex7, Spartan3E and Vertix7

Bits Size	Artix 7	Kintex 7	Spartan 3E	Virtex7
2	4690216	4677544	4499028	4677608
	4691240	4682408	4499156	4691048
	4691240	4682408	4499156	4691048
	4499028	4499028	4499028	4499028
	4691240	4682408	4499156	4691048
4	4691240	4690728	4500564	4690088
	4691240	4691432	4499988	4681960
	4690216	4677864	4499988	4677096
	4498004	4498004	4498004	4498004
	4689892	4676964	4498640	4690020
8	4691944	4691432	4506196	4678184
	4684968	4683048	4506324	4683048
	4678760	4691368	4510612	4676520
	4692504	4692504	4505284	4690968
	4690916	4691428	4691428	4676516
16	4692664	4693480	4527444	4505904
	4695528	4700072	4540692	4696104
	4681640	4684776	4526932	4693160
	4692056	4692504	4514436	4680920
	4683620	4682916	4530512	4694116

The data in table- I indicate that the memory usage is remained quite stable across different multiplier sizes and FPGA families, with values generally lying in the range of approximately 4.49×10^6 to 4.70×10^6 KB. For the 2-bit implementations, the memory requirement is stated to be around 4.50–4.69 million KB, while for 16-bit implementations it is reported to increase for several architectures, reaching approximately 4.70 million KB. Among the FPGA families, the lowest memory usage is generally shown by Spartan-3E, whereas slightly higher values are shown by Artix-7, Kintex-7, and Virtex-7. The moderately small variation suggests that the reported memory figure is influenced not only by multiplier size but also by the synthesis-tool environment and FPGA-specific implementation.

As the multiplier size is increased from 2-bit to 16-bit, a gradual increase in memory requirement can be observed for several multiplier architectures. However, the difference between individual algorithms is noted to be comparatively small compared with the overall reported memory value. This suggests that memory usage is less sensible to multiplier architecture than propagation delay, where significant differences were observed previously. For a complete comparative study, the memory results should therefore be analyzed together with delay, LUT utilization, number of logic levels, and power consumption to identify the most efficient multiplier–FPGA combination.

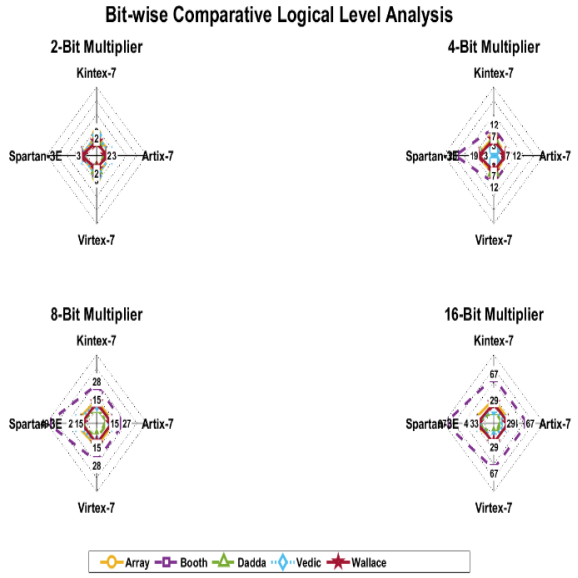

Fig. 2 Logical levels Analysis

The radar analysis as shown in figure 2, indicates that the logical levels increase with multiplier bit size across all FPGA families. For 2-bit multipliers, the differences among the architectures are relatively small; however, as the operand size increases to 4-, 8-, and 16-bit, the variation becomes more prominent. Dadda generally exhibits the lowest logical depth, followed by Vedic and Wallace, while Booth shows

the highest logical levels, particularly for 8- and 16-bit multipliers.

Across the FPGA families, Spartan-3E consistently shows higher logical levels compared with Artix-7, Kintex-7, and Virtex-7, with the difference becoming particularly significant for larger multipliers. Artix-7, Kintex-7, and Virtex-7 demonstrate comparatively similar trends. Overall, the results confirm that both multiplier architecture and FPGA technology influence logical depth, with Dadda providing a more efficient implementation for larger multiplier sizes.

The comparative analysis of resource utilization across Artix-7, Kintex-7, Spartan-3E, and Virtex-7 FPGAs in Table-II indicates that resource requirements generally increase with increasing bit size.

TABLE II. Comparative LUTs Analysis of Multiplier on Artix 7 , Kintex7, Spartan3E and Vertix7

Bits Size	Artix 7	Kintex 7	Spartan 3E	Virtex7
2	4	4	4	4
	5	5	12	5
	5	5	12	5
	4	4	4	4
	5	5	12	5
4	23	23	30	23
	57	57	83	57
	19	19	29	19
	3	4	4	4
	23	23	33	23
8	99	99	126	99
	235	235	290	308
	114	114	146	114
	121	121	192	121
	128	128	128	128
16	387	387	510	126
	834	834	1090	834
	453	453	623	453
	422	421	546	421
	499	499	682	499

For smaller bit-width designs, the utilization remains relatively low and comparable among the FPGA families. However, as the bit size increases, the resource consumption rises considerably, indicating increased hardware complexity and logic requirements.

Among the FPGA families, Artix-7 and Kintex-7 generally demonstrate better resource efficiency, particularly for larger designs, whereas Spartan-3E shows comparatively higher resource utilization in several configurations. Virtex-7 provides competitive results but exhibits variation depending on the design size and implementation. Overall, the results demonstrate that bit size and FPGA architecture significantly influence resource utilization, and selecting an appropriate FPGA family is essential for achieving an effective balance between hardware area and computational performance.

V. CONCLUSION AND FUTURE WORK

There is a clear trade-off between speed and resource utilization: Dadda/Wallace Multipliers, best choice when high speed (low delay) is the primary requirement. Vedic Multiplier, best choice when minimum FPGA resource utilization (low LUT count) is the primary objective. Array Multiplier, provides a balanced but not optimal solution in either speed or area. Therefore, for FPGA-based multiplier design: Choose Dadda or Wallace for performance-oriented applications. Choose Vedic for area-constrained applications where resource efficiency is more important than maximum speed.

Forthcoming work should include direct measurement or estimation of power consumption and throughput alongside delay and LUT utilization. The study can also be extended to higher operand widths and newer FPGA families. Hybrid architectures combining the parallelism of Vedic structures with compressor-tree reduction may further improve the speed–area–power trade-off for DSP, communication, and AI accelerator applications.

REFERENCES

[1] K. Mishra, E. Gupta, and A. Kumar, “Comparative research on power and junction temperature in 2, 4, 8 bit multiplier using Nikhilam Sutra,” in Proc. ICRASET, B. G. Nagara, Mandya, India, 2024, pp. 1–4, doi: 10.1109/ICRASET63057.2024.10895106.

[2] A. Kumar and R. P. Agarwal, “Performance evaluation and synthesis of FIR filters using various multipliers algorithms,” in Innovations in Electrical and Electronic Engineering, Lecture Notes in Electrical Engineering, vol. 756. Singapore: Springer, 2021, pp. [pages not provided], doi: 10.1007/978-981-16-0749-3_45.

[3] A. Kumar and R. P. Agarwal, "Design and implementation of efficient FIR low pass filters based on Vedic and traditional digital multiplier algorithm," in Advances in Mechanical Engineering, Lecture Notes in Mechanical Engineering. Singapore: Springer, 2021, pp. [pages not provided], doi: 10.1007/978-981-16-0942-8_25.

[4] A. Kumar, E. Gupta, R. P. Agarwal, R. Jain, and K. [surname not provided], "Comparative research for managing delay in signal processing via multipliers," in Proc. ICPEICES-2018, Delhi Technological University, India, 2018, pp. 1549–1554, doi: 10.1109/ICPEICES.2018.8897312.

[5] V. Sharma and A. Kumar, "Design, implementation & performance of Vedic multiplier for different bit lengths," International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 4, pp. 7912–7919, Apr. 2017.

[6] A. Kumar and Vishikha, "Comparative analysis of Vedic & array multiplier," International Journal of Electronics and Communication Engineering and Technology, vol. 8, no. 3, pp. 17–27, May–Jun. 2017.

[7] A. Kumar and R. P. Agarwal, "Complex multiplier: Implementation using efficient algorithms for signal processing application," International Journal of Recent Technology and Engineering, vol. 8, pp. 1235–1239, 2019, doi: 10.35940/ijrte.C5147.118419.

[8] A. Kumar and R. P. Agarwal, "Simulation and implementation of efficient binary multiplier circuits," in Proc. ICRMR-2018, AIT Conference Centre, Thailand, Nov. 23–24, 2018, pp. 1–6.

[9] P. K. Mishra, E. Gupta, and A. Kumar, "Design and analysis of Nikhilam based Vedic architecture for high speed arithmetic operations on FPGA," Indian Journal of Science and Technology, vol. 18, no. 30, pp. 2453–2459, 2025, doi: 10.17485/IJST/v18i30.1208.

[10] S. Banerjee, "Comparative analysis of Vedic and conventional multipliers," IET Circuits, Devices & Systems, vol. 15, no. 6, pp. 525–532, 2021, doi: 10.1049/cds2.12041.

[11] M. Maheswari and S. Prabha, "Optimized Vedic multiplier using reversible logic," Microprocessors and Microsystems, vol. 71, 2020.