IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

Scalable Modified Vedic Multiplier Using HSCG-SCG-CSLA for High-Speed FPGA Arithmetic

Dr. Javeed MD[1], Dr. Radha Krishna A.N[2], Kambalpalli Parameshwari[3]
Associate Professor, Dept. of ECE[1,2]
Dept. of ECE[3]
Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, Hyderabad, India.
Javeed.rahmanee@gmail.com, dr.radhakrishnaece@gmail.com, kparameshwari934@gmail.com

Abstract: *High-speed multiplication is a dominant requirement in digital signal processing, image processing, cryptography, and embedded arithmetic units. This work develops a scalable modified Vedic multiplier based on the Urdhva-Tiryagbhyam vertical-and-crosswise principle and replaces conventional carry-intensive final addition with a modified Half Sum Carry Generation–Sum Carry Generation Carry Select Adder (HSCG-SCG-CSLA). The architecture is constructed hierarchically from small Vedic cells, aligned partial products, and a decoder-assisted carry generation network. The design is described in Verilog HDL and evaluated using the Vivado simulation and synthesis flow for 4-, 8-, 16-, 32-, and 64-bit operands. The supplied implementation data show consistent LUT reduction for the multiplier family, including 20 versus 22 LUTs at 4 bits, 472 versus 482 LUTs at 16 bits, and 8147 versus 8311 LUTs at 64 bits. The delay result is most significant at 64 bits, where the proposed architecture reports 2.0371 ns compared with 3.6635 ns for the decoder-based Vedic baseline, corresponding to a 44.4% reduction. At 16 bits, delay decreases from 3.0507 ns to 2.8432 ns and total power from 45.541 W to 44.554 W. The results therefore indicate that the architectural benefit increases with selected word lengths, particularly when carry resolution dominates the critical path. The paper also discusses cases where the proposed structure introduces small overheads, emphasizing the importance of word-length-aware optimization.*

Keywords: Vedic multiplier, Urdhva-Tiryagbhyam, HSCG-SCG, CSLA, FPGA, Verilog HDL, Vivado, low-latency arithmetic

I. INTRODUCTION

Binary multiplication is a fundamental arithmetic operation in microprocessors, digital signal processors (DSPs), image-processing accelerators, cryptographic engines, and multiply-accumulate datapaths. In these systems, multiplier latency frequently becomes part of the critical timing path, while the large number of partial-product and carry-propagation nodes also contributes to logic utilization and switching power. Conventional shift-and-add, array, carry-save, Wallace-tree, Booth-recoded, and carry-select based architectures address different aspects of this problem, but their efficiency depends strongly on the word length and on the structure used for partial-product reduction.

The source design considered in this work combines two ideas. First, Urdhva-Tiryagbhyam multiplication produces partial products using a vertical-and-crosswise computation pattern that is naturally amenable to parallel hardware. Second, the final accumulation network is modified by using an HSCG-SCG-based carry-select adder. The objective is not merely to increase parallelism, but to reduce the time spent resolving carry information in the last arithmetic stages. The supplied project describes this approach as a modified Vedic multiplier suitable for Verilog HDL implementation and Vivado-based verification [1].

A key contribution of this paper is a publication-oriented reformulation of the supplied architecture and experimental results. Rather than assuming that every word length benefits identically, the reported synthesis tables are analyzed across 4, 8, 16, 32, and 64 bits. This reveals clear LUT savings across all evaluated multiplier sizes, strong delay improvement

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

at 16 and 64 bits, and small delay or power trade-offs at some smaller or intermediate widths. Such a word-length-aware interpretation is important when the multiplier is intended for practical FPGA datapaths.

II. RELATED WORK AND RESEARCH GAP

Vedic multiplication has been explored as an alternative to conventional binary multiplier structures because its partial products can be generated and combined in a regular hierarchy. The Urdhva-Tiryagbhyam principle is especially attractive for hardware because it supports simultaneous vertical and crosswise products and can be recursively extended from 2×2 cells to larger word lengths. Prior work has combined Vedic multiplication with carry-save adders, compressor trees, Wallace reduction, Booth encoding, barrel shifting, and modified carry-select structures to improve speed, power, or area. The supplied literature also reports the use of Vedic arithmetic in MAC units, FIR filters, QAM transmitters, and low-power DSP applications[2-5-.

A major limitation of many scalable multiplier implementations is that reduction of partial products does not completely eliminate the final carry-propagation bottleneck. Carry-select adders reduce this delay by preparing candidate sums, but conventional CSLA structures can require substantial multiplexing and duplicated logic. The HSCG-SCG-CSLA used in the source design attempts to reduce this overhead by generating half-sum/half-carry and final-sum/final-carry information through specialized stages and by arranging the stages in progressively sized groups. For a 16-bit SQRT-style adder, the supplied gate-count comparison reports 220 gates for the HSCG-SCG-CSLA, compared with 336 for BEC-SQRT-CSLA and 434 for regular CSLA [6-7].

The research gap is therefore a scalable multiplier architecture that preserves the parallel partial-product advantage of Vedic arithmetic while reducing the cost of the final addition network. The proposed method addresses this gap by replacing conventional parallel adders in the Vedic hierarchy with the HSCG-SCG-CSLA and by evaluating the resulting design over multiple operand widths.

III. PROPOSED MODIFIED VEDIC MULTIPLIER

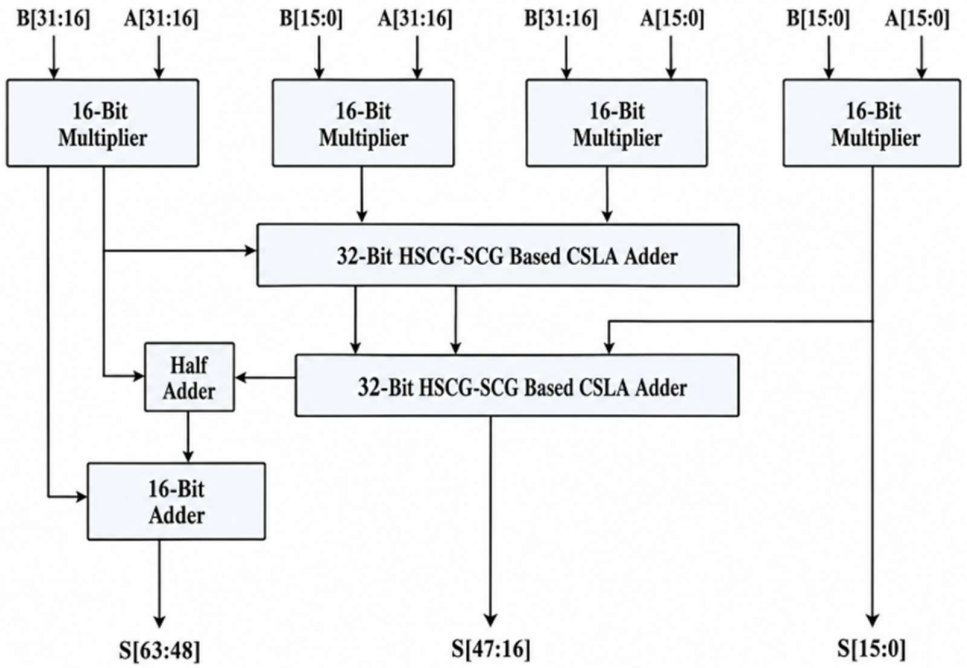

Fig. 1. Proposed scalable HSCG-SCG-based modified Vedic multiplier architecture.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

The proposed multiplier follows a hierarchical construction. An N-bit operand is divided into high and low halves. Four smaller Vedic multipliers operate on the combinations AH×BH, AH×BL, AL×BH, and AL×BL. Their outputs are shifted according to significance and combined by a hierarchical reduction network. At the last stage, the HSCG-SCG-CSLA resolves the aligned sum and carry vectors to produce the 2N-bit product. This recursive organization permits the same design strategy to be extended from 4-bit to 8-, 16-, 32-, 64-bit and larger multipliers.

3.1 Urdhva-Tiryagbhyam Partial-Product Generation

For the elementary 2×2 case, let A = a1a0 and B = b1b0. The least significant product bit is formed vertically, the cross products form the middle terms, and the most significant product is combined with the carry generated by the crosswise addition. The hardware interpretation is compact and is suitable for recursive construction.

$$p0 = a0 \cdot b0$$
$$(c1, p1) = a1 \cdot b0 + a0 \cdot b1$$
$$(c2, p2) = a1 \cdot b1 + c1, \quad P = \{c2, p2, p1, p0\}$$

The main advantage of the Vedic formulation is that the partial products can be generated concurrently. For higher word lengths, the smaller blocks operate in parallel and only the aligned intermediate terms need to be accumulated. The critical design choice therefore shifts from partial-product generation to efficient final addition.

3.2 Modified HSCG-SCG-CSLA

The HSCG-SCG full-adder stage is organized around half-sum generation (HSG), half-carry generation (HCG), final-sum generation (FSG), and final-carry generation (FCG). The HSG and HCG paths prepare intermediate sum and carry information from the operand bits. The FSG and FCG stages combine this information with Cin to obtain the final sum and carry. Group-M structures cascade M such stages, and progressively increasing group widths are used to construct the complete carry-select adder. The supplied design uses five groups for a 16-bit implementation.

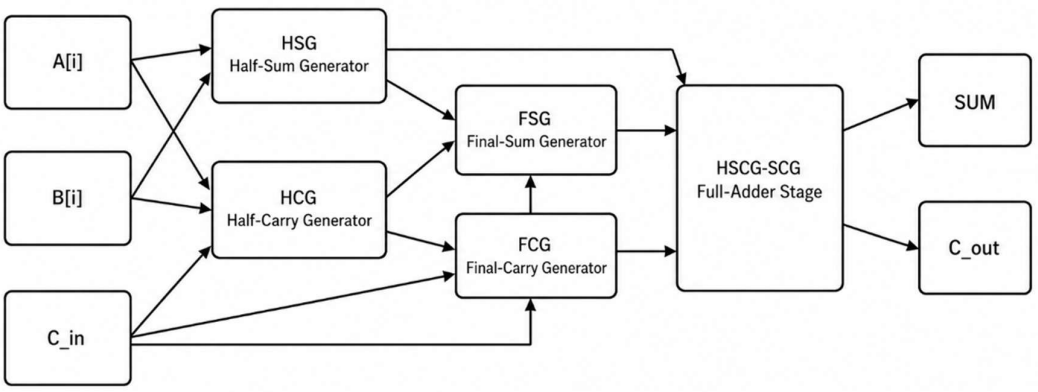

Fig. 2. Conceptual organization of the modified HSCG-SCG-CSLA stage and group hierarchy.

The motivation for the modified adder is visible in the supplied gate-count data. The 16-bit regular CSLA requires 434 gates, the BEC-SQRT-CSLA requires 336 gates, and the HSCG-SCG-CSLA requires 220 gates. Relative to the regular CSLA, this represents a 49.3% reduction in the reported gate count. The reduction is primarily associated with simplified group logic and fewer multiplexer-oriented carry-selection elements.

Adder structure	Group 1	Group 2	Group 3	Group 4	Group 5	Total gates
Regular CSLA	26	57	87	117	147	434
BEC-SQRT-CSLA	26	43	66	89	112	336

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

HSCG-SCG-CSLA	24	28	42	56	70	220

Table I. Reported gate-count comparison for 16-bit SQRT-style adders.

IV. RTL IMPLEMENTATION AND VERIFICATION METHODOLOGY

The architecture is described in Verilog HDL and follows a bottom-up reusable-module methodology. A 2×2 Vedic cell forms the base multiplication unit. Larger Vedic blocks are composed recursively, and the HSCG-SCG arithmetic stages are instantiated in the alignment and final-addition network. Parameterized word-length definitions allow the same structural concept to be evaluated at 4, 8, 16, 32, and 64 bits.

Functional verification is performed using simulation test vectors that exercise zero operands, maximum unsigned values, alternating-bit patterns, and representative random operand pairs. For each test, the RTL product is compared with the mathematical unsigned multiplication result. Synthesis in Vivado is then used to obtain LUT count, slice-register usage, bonded I/O count, power estimates, and combinational delay. The same flow is applied to the decoder-based Vedic baseline and the proposed HSCG-SCG-based multiplier so that the comparison is made under a common tool methodology.

The supplied project includes simulation outputs for 4-, 8-, 16-, 32-, and 64-bit proposed multipliers together with synthesized and RTL schematic views. The results reported below are therefore treated as tool-derived implementation data rather than as transistor-level silicon measurements. Power numbers are synthesis-tool estimates and should be interpreted comparatively within the same flow.

V. RESULTS AND DISCUSSION

Table II summarizes the reported multiplier implementation results. The proposed architecture reduces LUT usage at every evaluated word length. The improvement is modest at 8 and 32 bits but becomes more meaningful in absolute terms at 64 bits, where 164 LUTs are removed. Slice-register behavior is mixed: the proposed design uses fewer registers up to 32 bits, whereas the 64-bit result reports 2424 registers versus 2397 for the baseline.

Bits	**LUTs E/P**	**Slice regs E/P**	**Total power (W) E/P**	**Delay (ns) E/P**	**Delay improvement**
4	22/20	7/6	4.048/4.196	1.8155/1.8331	-0.97%
8	104/103	28/28	13.938/14.008	2.5409/2.4930	+1.89%
16	482/472	145/135	45.541/44.554	3.0507/2.8432	+6.80%
32	1988/1967	575/559	145.602/145.491	3.0177/3.1311	-3.76%
64	8311/8147	2397/2424	450.498/450.125	3.6635/2.0371	+44.39%

Table II. Existing (E) versus proposed (P) Vedic multiplier results from the supplied implementation data.

5.1 Timing Behaviour

The delay results show that the proposed architecture is most effective at word lengths where the final carry-resolution path dominates. At 16 bits, delay falls from 3.0507 ns to 2.8432 ns, a reduction of approximately 6.80%. At 64 bits, delay falls from 3.6635 ns to 2.0371 ns, corresponding to an approximately 44.4% reduction. The 8-bit design also improves by about 1.89%. In contrast, the 4-bit design is 0.97% slower and the 32-bit design is 3.76% slower than the supplied decoder-based baseline. These two cases indicate that the fixed overhead of the modified selection and reduction logic can outweigh its carry-path advantage at certain sizes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

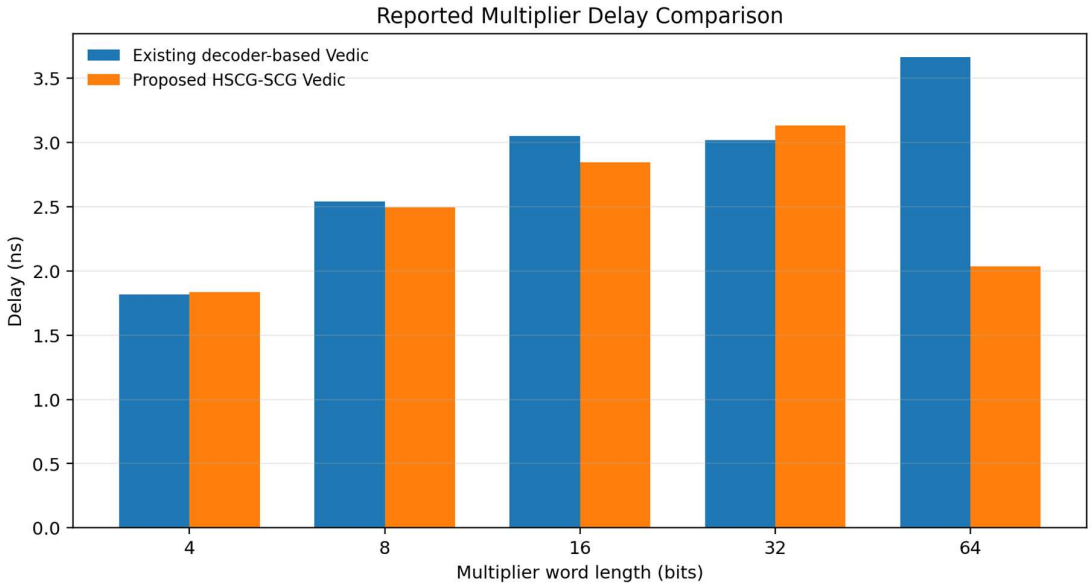

Fig. 3. Reported delay comparison across multiplier word lengths.

5.2 Area and Power Behaviour

LUT utilization is reduced for all multiplier sizes: from 22 to 20 at 4 bits, 104 to 103 at 8 bits, 482 to 472 at 16 bits, 1988 to 1967 at 32 bits, and 8311 to 8147 at 64 bits. This supports the design objective of limiting logic growth in the final accumulation stage. The associated LUT reduction is approximately 9.09%, 0.96%, 2.07%, 1.06%, and 1.97%, respectively.

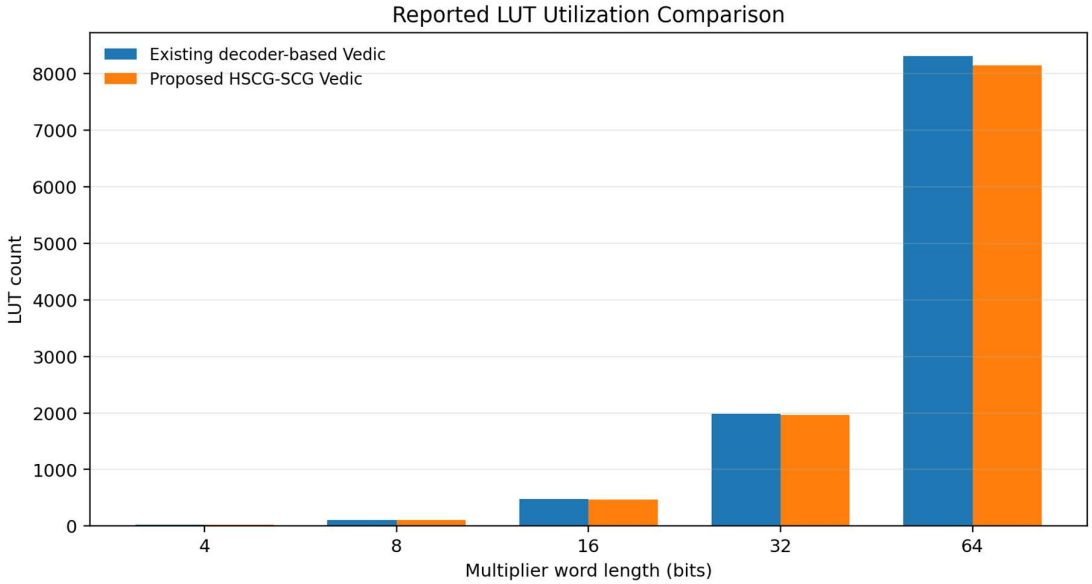

Fig. 4. Reported LUT utilization of existing and proposed multipliers.

The total power trend is more nuanced. The 16-bit multiplier reduces total power from 45.541 W to 44.554 W (about 2.17%), while the 32- and 64-bit cases show very small reductions. The 4- and 8-bit implementations show small

DOI: 10.48175/IJARSCT-38270

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

increases in total power. These results suggest that the primary demonstrated advantage of the proposed method is delay reduction and logic restructuring rather than universal power reduction at every word length. For FPGA deployment, the design should therefore be selected using a combined timing-area-power objective rather than a single metric.

5.3 Adder-Level Results

Adder width	LUTs E/P	Total power (W) E/P	Delay (ns) E/P
4	4/4	3.012/3.013	2.2090/2.1960
8	11/11	6.030/6.022	2.3430/2.3080
16	21/20	7.265/7.272	2.5190/2.5080
32	33/34	11.493/11.741	5.2600/5.7400
64	94/88	52.962/52.927	2.4900/2.4686

Table III. Reported decoder-based CSLA versus HSCG-SCG adder metrics.

At the adder level, the proposed HSCG-SCG structure slightly reduces delay at 4, 8, 16, and 64 bits, while the 32-bit result is slower. LUT count is identical at 4 and 8 bits, reduced at 16 and 64 bits, and increases by one LUT at 32 bits. These data reinforce the conclusion that the architecture should be synthesized and tuned per target width rather than assumed to scale monotonically.

VI. RESULTS

The modified Vedic multiplier is suitable for arithmetic-dense FPGA and VLSI applications in which multiplication is executed repeatedly and latency is important. Typical targets include FIR and IIR filters, FFT processing, image convolution, matrix arithmetic, cryptographic modular operations, digital modulation, MAC accelerators, and embedded AI preprocessing. The recursive architecture is attractive when designers require multiple precision options because the same small Vedic modules can be reused hierarchically.

Three implementation considerations follow from the reported results. First, synthesis constraints and FPGA family characteristics can change the mapping of the HSCG-SCG logic into LUTs and carry resources. Second, synthesis power estimates depend on assumed switching activity and should be validated with post-route activity or hardware measurements before making absolute low-power claims. Third, the 32-bit delay regression shows that placement, routing, and group partitioning can create a local critical-path penalty even when the logical gate count is reduced. Retiming, pipelining, or re-optimizing the group sizes may therefore improve intermediate word lengths.

VII. CONCLUSION AND FUTURE SCOPE

This paper presented a scalable modified Vedic multiplier that combines Urdhva-Tiryagbhyam partial-product generation with an HSCG-SCG-based carry-select final addition network. The approach is modular, Verilog-friendly, and suitable for FPGA implementation across multiple operand widths. The supplied synthesis data demonstrate LUT reduction at all evaluated multiplier sizes and substantial timing improvement at selected larger widths. In particular, the 64-bit proposed multiplier reduces reported delay from 3.6635 ns to 2.0371 ns while also lowering LUT count from 8311 to 8147. The 16-bit implementation provides a balanced result, reducing LUTs, slice registers, total power, and delay simultaneously. The results also show that improvement is not uniform: 4-bit and 32-bit delay increase slightly, and the smallest designs exhibit marginal power overhead. This is an important design insight because it identifies word-length-dependent overhead and motivates architecture tuning rather than indiscriminate scaling. Future work can include pipelining, optimized group partitioning, post-route timing closure, dynamic-power evaluation using realistic switching activity, signed/unsigned unified multiplication, radix-8 or radix-32 recoding, and hybrid partial-product compression. The

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, September 2026
Impact Factor: 8.2

supplied project further suggests reducing partial-product column height in modified Booth-recoded multipliers and extending the design toward pipelined multiplier structures [1].

REFERENCES

1. Kumari, S. Kharwar, S. Singh, M. K. A. Mohammed, and S. M. Zaki, "Design and Implementation of Modified Vedic Multiplier Using Modified HSCG-SCG Adder," in Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems, 2023.
2. V. Krishna, S. Deepthi, and M. N. Devi, "Design of 32-bit MAC unit using Vedic multiplier and XOR logic," in Proceedings of the International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications, pp. 715–723, Springer, 2021.
3. M. O'Connor and E. E. Swartzlander, "Exploiting asymmetry in Booth-encoded multipliers for reduced energy multiplication," in 49th Asilomar Conference on Signals, Systems and Computers, pp. 722–726, IEEE, 2015.
4. T. Srujana, D. Divya, and M. Javeed, "FPGA Implementation for Detection Disease on Cotton Plants Using Advanced Image Processing Algorithm," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 7, no. 11, pp. 4023–4030, Nov. 2018. Google Scholar
5. D. Tiwary and M. Javeed, "Power Management for Low Power VLSI Design," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 6, no. 2, pp. 732–735, Apr. 2018. Google Scholar
6. M. Senthil Kumar and M. Javeed, "Design of NoC Router with 3PE, Double and Triple Error Detection by Using Improved Hamming Code," *ARPN Journal of Engineering and Applied Sciences*, vol. 14, no. 16, pp. 2874–2880, Aug. 2019. Google Scholar
7. M. Javeed and M. Senthil Kumar, "Automatic Brain Tumour Detection Using New Structure Algorithm," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 11, pp. 3402–3405, Sep. 2019. Google Scholar