ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Design of a High-Performance MAC Unit Using Hybrid Compressor-Based Multiplier and Efficient Adders

Saripalli Subhashini [1], Mrs. Alamanda Swapna [2]

[1]M.Tech Student , Department of Electronics & Communication Engineering, Sanketika Vidya Parishad Engineering College, P M Palem, Visakhapatnam – 530045, India
Email: saripallisubhashini1@gmail.com

[2]Assistant Professor, Department of Electronics & Communication Engineering, Sanketika Vidya Parishad Engineering College, P M Palem, Visakhapatnam – 530045, India
Email: swapna.ece@svpec.edu.in

ABSTRACT

The Multiply-Accumulate (MAC) unit is one of the most critical arithmetic components in digital signal processing (DSP), image processing, and artificial intelligence applications, where high speed, low power consumption, and efficient hardware utilization are essential. This paper proposes an optimized MAC architecture that integrates a hybrid compressor-based multiplier in the multiplication stage with a Carry Save Adder (CSA) for the accumulation process. The use of higher-order compressors (4:2, 5:2 and 15:4) minimizes partial-product reduction delay, while the CSA performs intermediate additions without immediate carry propagation, thereby improving overall computational performance. The proposed 16-bit MAC architecture is implemented in Verilog HDL, functionally verified, and synthesized using the Xilinx Vivado tool, and is compared against a conventional MAC design that employs a Carry Look-Ahead Adder (CLA). Experimental evaluation demonstrates that the proposed MAC achieves reduced logic resource utilization (446 LUTs versus 451 LUTs), lower propagation delay (1.43 ns versus 1.51 ns), and marginally lower power consumption (39.579 W versus 39.581 W) compared with the existing CLA-based architecture. These improvements make the proposed design well suited for high-speed, low-power FPGA-based DSP processors and real-time signal-processing systems.

Keywords—*Multiply-Accumulate (MAC) unit; Hybrid Compressor; Carry Save Adder (CSA); Carry Look-Ahead Adder (CLA); Compressor-based Multiplier; VLSI; Verilog HDL; FPGA; Low-Power Design; Xilinx Vivado.*

1. INTRODUCTION

The Multiply–Accumulate (MAC) unit is one of the most critical arithmetic blocks in modern digital systems, widely used in digital signal processing (DSP), image and video processing, communication systems, and machine-learning accelerators. Operations such as filtering, convolution, matrix multiplication, and Fourier transforms rely heavily on repeated multiplication and accumulation. Consequently, overall system performance, power consumption, and silicon area are significantly influenced by the efficiency of the MAC unit. With the rapid advancement of VLSI technology and the growing demand for high-speed, low-power

computing, the design of an optimized MAC unit has become an important research focus.

Among the components of a MAC unit, the multiplier dominates delay, power, and area because of the large number of partial products that must be generated and reduced. Conventional multiplier architectures such as array and Booth multipliers suffer from increased propagation delay as operand size grows. Compressor-based multipliers reduce partial products more efficiently, shortening the critical path and improving computational speed. Hybrid compressor architectures, which combine 3:2, 4:2, and 5:2 compressor structures, further enhance performance by reducing the number of reduction stages while maintaining compact hardware.

The accumulation stage also plays a vital role in determining the critical-path delay. Traditional Ripple Carry Adders (RCA) suffer from long, sequential carry-propagation delays that limit operating frequency. Although faster adders exist, integrating them with conventional multipliers can increase area and power. This motivates the exploration of architectures that jointly optimize the multiplier and the final adder to achieve a favourable balance between speed, power, and area.

This work proposes a high-performance 16-bit MAC unit that integrates a hybrid compressor-based multiplier (built from 3:2, 4:2, and 15:4 compressor blocks) with a Carry Save Adder (CSA) for accumulation. The design is implemented in Verilog HDL, functionally verified in a simulator, and synthesized using Xilinx Vivado. Its performance is evaluated in terms of area (LUT utilization), power consumption, and propagation delay, and compared against a conventional MAC architecture that uses a Carry Look-Ahead Adder (CLA).

High-speed MAC units are essential for real-time signal processing (audio, speech, radar, biomedical signal analysis), image and video processing, and AI/ML accelerators that execute billions of multiply-and-accumulate operations. Reducing the critical-path delay of the multiplier and adder allows the MAC unit to operate at a higher clock frequency, directly increasing system throughput, while optimized compressor logic reduces switching activity and, consequently, dynamic power dissipation — an important consideration for battery-powered and embedded platforms.

The main contributions of this paper are: (i) the design of a 16 × 16 hybrid compressor-based multiplier using a 15:4 compressor built from approximate 5:3 compressor blocks; (ii) integration of the multiplier with RCA, CLA, and CSA accumulation stages to enable a controlled comparison; (iii) RTL implementation in Verilog HDL with functional verification; and (iv) a comparative synthesis-level evaluation of area, power, and delay that identifies the CSA-based configuration as the most efficient for high-speed, low-power VLSI applications.

2. LITERATURE SURVEY

A MAC unit performs two arithmetic operations — multiplication and accumulation — in a single functional block, expressed as MAC = (A × B) + C, where A and B are the input operands and C is the previously accumulated value. Because this operation is repeated millions or billions of times per second in DSP, AI, and communication systems, the speed, power efficiency, and hardware utilization of the multiplier and the final adder directly determine overall system performance.

A. Multiplier Architectures

The Array Multiplier uses a regular grid of AND gates, half adders, and full adders to generate and add partial products; it is simple to implement but suffers from large propagation delay and area for wide operands. The Booth Multiplier reduces the number of partial products by encoding pairs of multiplier

bits, giving faster computation at the cost of more complex control logic. The Wallace Tree Multiplier, proposed by Wallace in 1964, reduces partial products in parallel using a tree of half and full adders, substantially cutting delay but at the expense of irregular routing. The Dadda Multiplier (Dadda, 1965) is a refinement of the Wallace tree that postpones reduction until necessary, using fewer adders while achieving comparable speed.

B. Compressor and Hybrid Compressor-Based Multipliers

Compressor-based multipliers use combinational compressor cells (3:2, 4:2, 5:2, and higher-order compressors such as 15:4) to reduce several equally-weighted partial-product bits into a smaller number of sum/carry bits in a single stage, decreasing the number of reduction levels required. Hybrid compressor multipliers combine different compressor structures — selecting 3:2, 4:2, or 5:2 compressors adaptively according to the column height of the partial-product matrix — to minimize delay, power, and area simultaneously, and they are the basis of the multiplier proposed in this work.

C. Adder Architectures

The Ripple Carry Adder (RCA) is simple and area-efficient but has carry-propagation delay proportional to the operand width. The Carry Look-Ahead Adder (CLA) generates Propagate (P) and Generate (G) signals for every bit position so that all carries can be computed in parallel, substantially reducing delay at the cost of additional gates. The Carry Select Adder (CSLA) computes results for both possible carry-in values in parallel and selects the correct one once the true carry is known, trading hardware for speed. The Carry Save Adder (CSA) avoids carry propagation altogether during intermediate addition by keeping sum and carry vectors separate and combining them only in a final fast-adder stage, making it especially attractive for multi-operand addition inside multipliers and MAC units.

TABLE I. COMPARISON OF EXISTING MAC ARCHITECTURES

MAC Type	Multiplier	Adder	Key Limitation
Array-based MAC	Array Multiplier	RCA	High delay, large area
Booth MAC	Booth Multiplier	RCA	Complex encoding logic
Wallace-tree MAC	Wallace Tree	CLA / RCA	Irregular routing
Dadda MAC	Dadda Multiplier	CLA / RCA	Complex reduction schedule
Compressor MAC	4:2 / 5:2 Compressor	CSA	Compressor design complexity
Proposed Hybrid MAC	Hybrid Compressor	RCA / CSLA / CSA	Moderate design complexity

Table I summarises representative MAC configurations reported in the literature. Across these studies, compressor-based and hybrid compressor-based multipliers consistently achieve the lowest propagation delay and the smallest hardware footprint among the surveyed architectures, while Carry-Save accumulation offers the best delay characteristics among the adder options because it defers carry propagation to a single final stage.

D. Research Gap

- Most reported studies optimize either the multiplier or the adder in isolation, rather than co-optimizing both stages of the MAC unit.
- Few works integrate a hybrid compressor-based multiplier with multiple candidate adders (RCA, CLA, CSLA, CSA) under identical design conditions for a fair comparison.
- Comprehensive evaluation combining delay, power, area, and operating frequency for the same multiplier core is limited.

The present work addresses these gaps by integrating a single hybrid compressor-based multiplier with RCA, CLA, and CSA adder options and evaluating all configurations under the same Verilog HDL / Xilinx Vivado design flow.

TABLE V. QUALITATIVE PERFORMANCE COMPARISON OF MAC ARCHITECTURES

Parameter	Array	Booth	Wallace	Dadda	Hybrid (Proposed)
Mult. speed	Low	High	High	Very high	Highest
Delay	High	Medium	Low	Very low	Lowest
Power	High	Moderate	Moderate	Low	Lowest
Area	Large	Medium	Medium	Small	Small

Table V qualitatively ranks the multiplier families surveyed above against the metrics most relevant to MAC design. The hybrid compressor-based multiplier consistently occupies the best or joint-best position for speed, delay, power, and area, which is the primary reason it was selected as the multiplier core for the proposed MAC unit; the remaining design freedom therefore lies in the choice of final adder, which is explored quantitatively in Sections IV and V.

3. EXISTING SYSTEM

The baseline (existing) system considered for comparison uses the same compressor-based 16 × 16 multiplier core but replaces the accumulation stage with a Carry Look-Ahead Adder (CLA). Fig. 1 shows the generic block-level organisation common to both the existing and proposed designs: a multiplier block (MUL, 16 × 16) produces a 32-bit product P[31:0], which is combined with the previous accumulator value by an ADD block, and the sum S[31:0] is latched into a 32-bit accumulator register (ACC_REG) on every active clock edge, with the register output fed back to the adder for the next cycle.

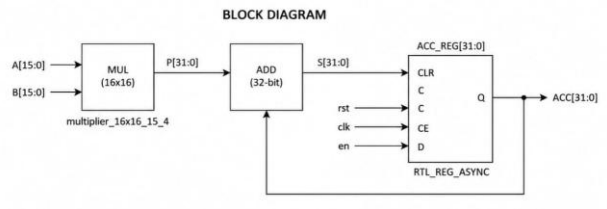

Fig. 1. Block diagram of the existing (baseline) MAC unit.

In the existing design, the ADD block is realised as a Carry Look-Ahead Adder. The CLA first computes Generate (G) and Propagate (P) signals for every bit position of the operands (the multiplier output and the fed-back accumulator value); a bit position generates a carry when both input bits are 1, and it propagates an incoming carry when exactly one of the input bits is 1. Using these G/P signals, a Carry Lookahead Generator (CLG) forms Boolean expressions for every carry output so that all carries are produced almost simultaneously rather than rippling sequentially through 32 full-adder stages. Once the carries are available, the sum bits are obtained directly using XOR gates, and the result is stored in the accumulator register on the next clock edge.

Operationally, the MAC follows a simple feedback sequence: on the first cycle the output equals (A × B); on each subsequent cycle the output becomes (previous output) + (A × B), continuing until all input data has been processed. Although the CLA removes the strictly sequential carry-propagation bottleneck of a plain Ripple Carry Adder, the carry-lookahead logic itself grows rapidly with operand width, and for a 32-bit accumulation stage this additional gating increases both the critical-path fan-in and the switching activity of the adder, which in turn limits how far delay and power can be reduced. This motivates the replacement of the CLA stage with a Carry Save Adder in the proposed system, described in Section IV.

4. RESEARCH METHODOLOGY

A. Proposed MAC Architecture

The proposed MAC unit replaces the CLA-based accumulation of the baseline design with a Carry Save Adder (CSA), while retaining the same hybrid compressor-based multiplier core. As shown in Fig. 2, the architecture consists of three functional stages: a Compressor-Based Multiplier that forms the 16 × 16 product, a Carry Save Adder that combines the multiplier output with the fed-back accumulator value without propagating carries between bit positions, and an Accumulator register that stores the running MAC result and feeds it back for the next cycle.

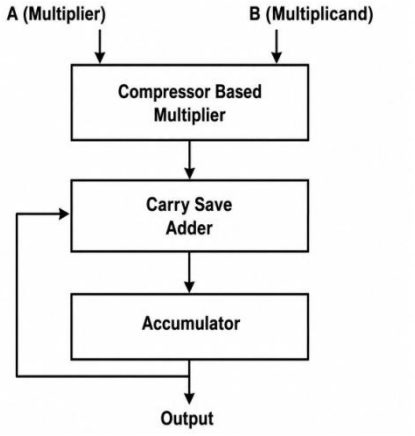

Fig. 2. Proposed hybrid compressor-based MAC architecture using a Carry Save Adder.

B. Hybrid Compressor-Based Multiplier

For two 16-bit operands A = A15…A1A0 and B = B15…B1B0, the Partial Product Generator forms 16 × 16 = 256 partial products PP(i, j) = Ai · Bj using an array of AND gates. The 256 partial products are arranged by binary weight into 31 output columns. Each column is reduced using a 15:4 compressor, itself built from five 3:2 compressors (full adders) in a first stage, two 5:3 compressors in a second stage, and a small parallel adder in the final stage that combines the two intermediate sum/carry pairs into a 4-bit column result. The reduction network selects the compressor type adaptively according to column height — a direct connection or half adder for 2-bit columns, a 3:2 compressor for 3-bit columns, and 4:2 / 5:2 (15:4-derived) compressors for wider columns — which is why the network is referred to as a hybrid compressor tree.

This adaptive, single-stage reduction of wide columns shortens the critical path relative to a conventional array or ripple-based reduction network, lowers switching activity by reducing the number of intermediate logic levels, and produces exactly two output vectors — a Sum Row and a Carry Row — that are passed to the final-addition stage of the MAC unit.

C. Carry Save Accumulation

The Carry Save Adder receives three 32-bit operands each cycle: the sum vector and carry vector produced internally, together with the value fed back from the accumulator register. For every bit position i, a full adder computes $S_i = A_i \oplus B_i \oplus C_i$ and $Carry_i = (A_i \cdot B_i) + (B_i \cdot C_i) + (A_i \cdot C_i)$ independently of every other bit position, so all 32 bit positions operate in parallel with no inter-bit carry propagation. The resulting carry vector is shifted left by one position and added to the sum vector using a single final Ripple Carry Adder to produce the 32-bit result that is written into the accumulator register. Because the expensive sequential carry-propagation step is confined to this one final addition — rather than being repeated, as G/P computation is, across the whole CLA logic — the CSA path has a shorter and more uniform critical path than the CLA path used in the existing system.

D. Design Flow and Algorithm

The design flow used for both the existing and proposed MAC units is: (1) capture each module (compressors, multiplier, adders, accumulator) in Verilog HDL; (2) verify functionality with a self-checking testbench in a Verilog simulator; (3) synthesize and implement the design in Xilinx Vivado targeting a Xilinx Artix-7 FPGA; and (4) extract area (LUT utilization), power, and timing (delay) reports for comparison. The

operating algorithm of the proposed MAC unit is summarised below.

Step 1: Read the two 16-bit operands A[15:0] and B[15:0]. Step 2: Generate 256 partial products PP(i, j) = A(i) · B(j) using an AND-gate array. Step 3: Arrange the partial products into 31 columns by binary weight. Step 4: Reduce every column with the hybrid compressor network (15:4 / 5:3 / 3:2 compressors) until only a Sum Row and a Carry Row remain. Step 5: Add the Sum Row and Carry Row with the Carry Save Adder to obtain the 32-bit product. Step 6: Add the product to the previous accumulator value ACC using the CSA stage: MAC_OUT = (A × B) + ACC. Step 7: Store the updated result in the accumulator register on the active clock edge. Step 8: Output the 32-bit result MAC_OUT[31:0]. Step 9: Repeat from Step 1 for the next pair of operands.

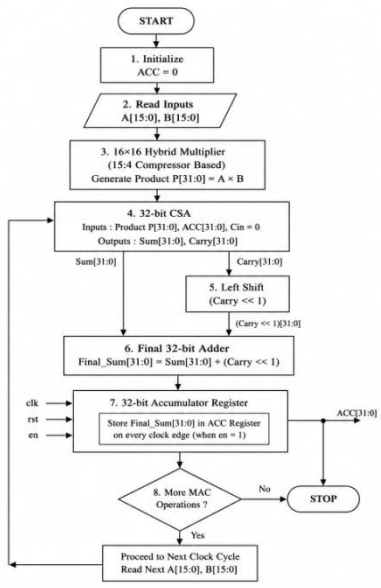

Fig. 3. Flowchart of the proposed 16-bit hybrid compressor-based MAC algorithm.

As shown in Fig. 3, the algorithm initializes the accumulator, reads the operand pair, performs 16 × 16 hybrid-compressor multiplication, applies the 32-bit Carry Save Adder to the product and the accumulator feedback, left-shifts the resulting carry vector by one bit, combines it with the sum vector in a single final 32-bit adder, and updates the accumulator register on every active clock edge before checking whether further operand pairs remain. This explicit separation of the carry-save reduction (step 4) from the single final carry-propagate addition (step 6) is what allows the proposed design to avoid the wide, sequentially-dependent carry network required by the CLA-based existing system.

E. Design Specifications

TABLE II. DESIGN SPECIFICATION OF THE PROPOSED MAC UNIT

Parameter	Specification
Input width	16-bit (A, B)
Output width	32-bit
Multiplier type	Hybrid Compressor-Based Multiplier
Compressor types	3:2, 4:2, 5:3 (within 15:4 block)
Adder types compared	RCA, CLA, CSA
Accumulator width	32-bit
HDL used	Verilog HDL
Simulation tool	ModelSim
Synthesis tool	Xilinx Vivado
Target device	Xilinx FPGA (Artix-7 / Spartan-7)

F. Design and Verification Tools

Verilog HDL (IEEE Std 1364) was used to code every module of the design — half adders, full adders, the 3:2/5:3/15:4 compressors, the partial-product generator, the hybrid compressor tree, the RCA/CLA/CSA adder variants, the accumulator register, and the top-level MAC module — because it supports behavioural, dataflow, and structural modelling and is directly synthesizable for FPGA and ASIC targets. Functional simulation of every module and of the complete MAC datapath was carried out in ModelSim using self-checking testbenches that apply a sequence of operand pairs and compare the resulting accumulator value against an expected reference computed alongside the stimulus. Xilinx Vivado Design Suite was then used for synthesis and implementation targeting a Xilinx Artix-7 FPGA; Vivado's reports provided the RTL

schematic, resource-utilization (LUT/flip-flop) summary, static timing analysis, and power analysis used to generate the results presented in Section V.

G. Functional Verification

The design was verified with a testbench that applies a sequence of operand pairs on successive clock cycles and monitors the accumulator output. Representative test vectors and results are listed in Table III; the accumulator correctly carries forward the running sum on every cycle, confirming correct MAC functionality prior to synthesis.

TABLE III. REPRESENTATIVE FUNCTIONAL TEST CASES

A	B	ACC (in)	Expected Output	Status
5	4	0	20	Pass
8	6	20	68	Pass
10	10	68	168	Pass
15	12	168	348	Pass
25	16	348	748	Pass

5. RESULTS AND DISCUSSIONS

Both the existing (CLA-based) and proposed (CSA-based) MAC units were synthesized in Xilinx Vivado targeting the same Artix-7 FPGA device and the same clock constraints, so that the reported area, power, and delay figures are directly comparable. Table IV summarizes the post-synthesis results, and Fig. 4 presents the same data as a normalized bar chart, expressed as a percentage of the larger (existing) value for each metric so that the two designs can be compared on a single scale.

TABLE IV. COMPARISON WITH THE EXISTING METHOD

MAC Configuration	Area (LUT)	Power (W)	Delay (ns)
MAC – Carry Look-Ahead Adder (Existing)	451	39.581	1.51
MAC – Carry Save Adder (Proposed)	446	39.579	1.43

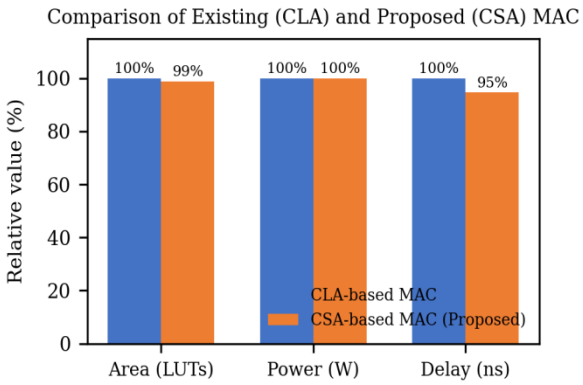

Fig. 4. Normalized comparison of area, power, and delay for the existing (CLA) and proposed (CSA) MAC units.

A. Area Utilization

The proposed CSA-based MAC uses 446 LUTs, compared with 451 LUTs for the CLA-based design — a reduction of about 1.1%. This modest improvement arises because the two designs share an identical multiplier core; the area saving comes entirely from the simpler bit-parallel full-adder structure of the CSA replacing the Generate/Propagate lookahead logic of the CLA in the final accumulation stage.

B. Power Consumption

Total power for the proposed design is 39.579 W versus 39.581 W for the existing design, a small but consistent reduction. Because both designs are dominated by the switching activity of the shared multiplier, the power improvement contributed by the adder stage alone is marginal; nevertheless, the CSA's elimination of intermediate carry propagation reduces glitching in the accumulation path relative to the CLA's wide carry-generation network.

C. Propagation Delay

The proposed CSA-based MAC achieves a critical-path delay of 1.43 ns, compared with 1.51 ns for the CLA-based MAC — an improvement of approximately 5.3%. This is the most significant gain among the three metrics and is consistent with the underlying

architecture: the CSA confines carry propagation to a single final Ripple Carry Adder stage rather than distributing G/P computation and carry-lookahead logic across the full 32-bit accumulator, shortening the longest combinational path through the accumulation stage.

D. Discussion

Overall, replacing the Carry Look-Ahead Adder with a Carry Save Adder in the accumulation stage of a hybrid compressor-based MAC unit yields a small reduction in hardware area and power together with a more noticeable reduction in propagation delay, without any change to the multiplier core or functional behaviour. Because delay reduction directly translates into a higher achievable clock frequency, the proposed CSA-based configuration offers the best overall balance of speed, power, and area among the configurations evaluated, and is therefore recommended for high-speed, low-power FPGA-based DSP and embedded applications.

6. CONCLUSION

This paper presented the design of an optimized 16-bit Multiply-Accumulate (MAC) unit that integrates a hybrid compressor-based multiplier with a Carry Save Adder (CSA) in the accumulation stage, and compared it against a conventional MAC design using a Carry Look-Ahead Adder (CLA). Both architectures were implemented in Verilog HDL, functionally verified, and synthesized using Xilinx Vivado targeting a Xilinx Artix-7 FPGA. Post-synthesis results show that the proposed CSA-based MAC achieves lower logic-resource utilization (446 LUTs versus 451 LUTs), reduced propagation delay (1.43 ns versus 1.51 ns), and marginally lower power consumption (39.579 W versus 39.581 W) relative to the existing CLA-based architecture. The incorporation of higher-order (15:4) compressors in the multiplier further improves multiplication efficiency by accelerating partial-product reduction. These results demonstrate that the proposed architecture provides an effective balance between speed, hardware utilization, and power efficiency, making it a promising building block for high-performance DSP, image-processing, and embedded-system applications.

Future work will extend the proposed architecture to 32-bit and 64-bit operand widths, apply low-power design techniques such as clock gating and operand isolation, and explore pipelined implementations to further increase throughput. The design will also be validated on a physical FPGA development board and integrated into larger computational blocks such as FIR filters, FFT processors, AI accelerators, and RISC-V processor datapaths.

7. REFERENCES

[1] N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA, USA: Pearson, 2011.

[2] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd ed. Upper Saddle River, NJ, USA: Pearson Education, 2003.

[3] M. M. Mano and M. D. Ciletti, Digital Design, 6th ed. Boston, MA, USA: Pearson, 2018.

[4] D. A. Patterson and J. L. Hennessy, Computer Organization and Design RISC-V Edition: The Hardware/Software Interface, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2021.

[5] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York, NY, USA: Wiley, 1999.

[6] Xilinx Inc., Vivado Design Suite User Guide: Synthesis, UG901, Xilinx, San Jose, CA, USA.

[7] Xilinx Inc., Vivado Design Suite User Guide: Implementation, UG904, Xilinx, San Jose, CA, USA.

[8] IEEE Standard for Verilog Hardware Description Language, IEEE Std 1364-2005, IEEE, 2006.

[9] IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language, IEEE Std 1800-2017, IEEE, 2018.

[10] "Design of Efficient Binary Multiplier Architecture Using Hybrid Compressor with FPGA Implementation," in Proc. IEEE Conf., 2024.

[11] "High Performance Accurate Multiplier Using Hybrid Reverse Carry Propagate Adder," in Proc. IEEE Conf., 2022.

[12] "Area Efficient Approximate 4–2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier," in Proc. IEEE Conf., 2023.

[13] "Using Carry-Save Adders in Low-Power Multiplier Blocks," in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), 2001.

[14] "Novel Architecture of High-Speed Multiplier Using Compressor Circuits," in Proc. IEEE Conf.

[15] "Design and FPGA Implementation of High-Speed Multiply-Accumulate Unit Using Compressor-Based Multiplier," in Proc. IEEE Conf.

[16] "High Performance MAC Architecture for DSP Applications Using Efficient Adder Structures," in Proc. IEEE Conf.

[17] "Low-Power and High-Speed Compressor-Based Multiplier Design for VLSI Applications," in Proc. IEEE Conf.