2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Efficient Booth Multiplier for FIR Filter Structure

Jeet Gandhi[1], Deepak Nair[1], Nehal Shah[1], Niket Singla[2]
[1]Sarvajanik College of Engineering and Technology, Surat.
[2]Sardar Vallabhbhai National Institute of Technology, Surat.

Abstract- In digital circuits for signal processing applications, multipliers play a crucial role in determining PPA (Power, Performance, and Area). It is difficult to strike a balance between these factors because improving one can have an adverse effect on others, resulting in inefficiencies. It is essential to multiply efficiently while using low power, minimal area, and high speeds. The Radix-4, Radix-8, and Radix-16 Booth Multiplier architectures are examined in this paper with an emphasis on minimizing partial products to lower computational complexity. In order to provide a more effective solution for contemporary circuit design, this paper suggests an optimized Radix-4 Booth Multiplier with improved PPA.

Keywords- radix-4, radix-8, radix-16 Booth Multiplier

1. Introduction-

Efficient digital filters are critical in modern signal processing applications such as wireless communication, active noise control, and embedded systems. These filters rely heavily on multipliers and adders, which dominate the computational load in operations like convolution and adaptive filtering. Particularly in devices like True Wireless Stereo (TWS) chips, where 128-tap FIR and LMS filters are common, optimized arithmetic units are essential to meet stringent power, area, and performance (PPA) requirements.
Multipliers significantly influence the overall design efficiency of FIR filters. Optimizing them is challenging due to the inherent trade-offs among power, area, and delay. Booth multipliers—especially those based on Radix-4, Radix-8, and Radix-16 encoding—reduce the number of partial products, thus lowering computational complexity.
This paper presents a **common parallel partial product generation (CPPG) method** implemented uniformly across **Radix-4, Radix-8, and Radix-16 Booth Multipliers**, aiming to minimize power and area while achieving high-speed performance. The unified CPPG architecture simplifies partial product computation and enables effective pipelining and hardware reuse. The proposed designs are evaluated using ASIC synthesis metrics, and their performance is compared with conventional architectures in terms of area, power, and delay. Additionally, the impact of these multiplier optimizations is assessed within FIR filter structures.

2. Concept of Booth Multiplier

Booth's algorithm is well-suited for multiplying signed numbers which are represented using two's complement. The algorithm reduces the number of partial products required when the multiplier contains consecutive 1's or 0's leading towards fewer additions. Booth multiplier can be designed using the components like (1) Shift Registers - For storing the multiplier and the accumulated result, (2) Adders/Subtractors - For adding or subtracting the multiplicand based on the bit pairs, (3) Control Logic - To determine when to add, subtract, or do nothing, based on the values of the relevant bits, and (4) Sign Extension - To ensure proper handling of signed numbers when negative values are involved.

2.1 Radix-4 Booth Multiplier

Radix-4 Booth's Algorithm is more efficient version compared to traditional Booth algorithm, where two bits of the multiplier are processed at a time, reducing the number of partial products further.

Radix-4 booth multiplier uses 3-bit encoding scheme at a time to reduce the number of partial products. It reduces the number of partial products approximately by half compared to a simple Radix-2 multiplier hence efficient for hardware implementation in terms of speed and area. For Radix-4 booth multiplier, possible partial products are $0, \pm b$, and $\pm 2b$ as indicated in table 1.

Partial product= $\frac{Number\ of\ multiplier\ bits}{2}$ (1)

The Radix-4 architecture in [6] reduces partial product generation through an optimized Booth encoding scheme that integrates sign selection and partial product generation using multiplexers, minimizing redundant logic and delay. Signed compressors are employed for efficient two's complement handling without incurring power, area, or delay overhead. The final summation stage uses an optimized carry look-ahead adder. Overall, this architecture achieves significant reductions in delay, power, area, and combined PPA metrics such as PDP and ADP.

2.2 Radix-8 Booth Multiplier

Radix-8 booth multiplier uses 4-bit encoding scheme at a time to reduce the number of partial products.

Partial product= $\frac{Number\ of\ multiplier\ bits}{3}$ (2)

Reduces the number of partial products approximately by one third compared to a simple binary multiplier. Efficient for hardware implementation in terms of speed and area. This introduces the possible partial products are 0, $\pm b$, $\pm 2b$, $\pm 3b$, $\pm 4b$.

The Radix-8 Booth Multiplier architecture [7] reduces redundant computations using XOR-XNOR logic and pre-computes operand magnitudes via MUX and binary-to-complement (B2C) logic. It efficiently handles non-trivial values using pre-computation to minimize partial products. This design improves speed and energy efficiency, especially for wider operands, and offers advantages over conventional Booth multipliers in terms of power, critical path delay (CPD), and design cost.

2.3 Radix-16 Booth Multiplier

Radix-16 booth multiplier uses 5-bit encoding scheme at a time to reduce the number of partial products.

Partial product= $\frac{Number\ of\ multiplier\ bits}{4}$ (3)

Reduces the number of partial products approximately by one fourth compared to a simple binary multiplier. Efficient for hardware implementation in terms of speed and area. This introduces the possible partial products are 0, $\pm b$, $\pm 2b$, $\pm 3b$, $\pm 4b$, $\pm 5b$, $\pm 6b$, $\pm 7b$, $\pm 8b$.

The Radix-16 Booth Multiplier architecture [8] utilizes pipelining and multiplexer-based logic to generate partial products (PPs) through shift-and-add operations based on Booth-encoded digits. By minimizing the number of Booth digit combinations, the design enhances computational efficiency. The architecture is optimized with a focus on reducing area, power, energy, and delay to achieve better performance than Radix-8 and higher-order Booth multipliers.

Table 1 Comparison of Booth Encoding Schemes for Radix-4, Radix-8, and Radix-16 Multipliers

Scheme	Group Width	Encoding bits	Partial Products	Operations
Radix-4	3 bits	[000, 001, … 110, 111]	8	0, ±A, ±2A
Radix-8	4 bits	[0000, 0001, …, 1110, 1111]	16	0, ±A, ±2A, ±3A, ±4A
Radix-16	5 bits	[00000, 00001, …, 11110, 11111]	32	0 to ±8A

Table 2 comparison of Multipliers Circuit [6], [7], [8]

Design	Radix	Frequency (MHz)	Area $(um)^2$	Delay (ns)	Power (mW)
[6]	4	1111.11	539.44	0.856	1.07
[7]	8	476.2	1875	1.69	0.46
[8]	16	729	350	2.74	54

A comparative analysis of Radix-4, Radix-8, and Radix-16 multiplier architectures was performed based on frequency, area, delay, and power. Radix-4 achieved the highest frequency (1111.11 MHz), lowest delay (0.856 ns), and moderate area (539.44 µm²) with 1.07 mW power consumption. Radix-8 showed reduced power (0.46 mW) but at the cost of increased area (1875 µm²) and delay (1.69 ns). Radix-16 offered the smallest area (350 µm²) but suffered from high delay (2.74 ns) and excessive power usage (54 mW). These results highlight a trade-off: Radix-4 is optimal for speed, Radix-8 balances area and power, while Radix-16, despite area efficiency, is less suited for low-power, high-speed applications.

3. Proposed Design

3.1 Optimized Computation Logic in Proposed design

The proposed optimization involves precomputing the partial products for the multiplicands *+b*, *+2b*, *-b*, and *-2b* in parallel with the Booth encoding scheme. Instead of dynamically generating partial products during multiplication, the required values are selected based on the encoded bits. This approach significantly reduces the area required for partial product generation.

For example, in an 8-bit multiplier, the standard method requires an area proportional to 4×area, whereas the optimized approach reduces it to *1×area*. Similarly, for a 64-bit multiplier, the standard approach demands *32×area*, while the optimized method requires only *1×area* at the partial product generation stage.

Additional optimizations are applied to arithmetic shift operations and binary-to-two's complement conversions by adopting alternative methods that enhance power efficiency, performance, and area utilization. Furthermore, the use of high-speed adder architectures, such as the Carry Look-Ahead Adder (CLA), although slightly increasing the area, provides substantial improvements in computational speed.

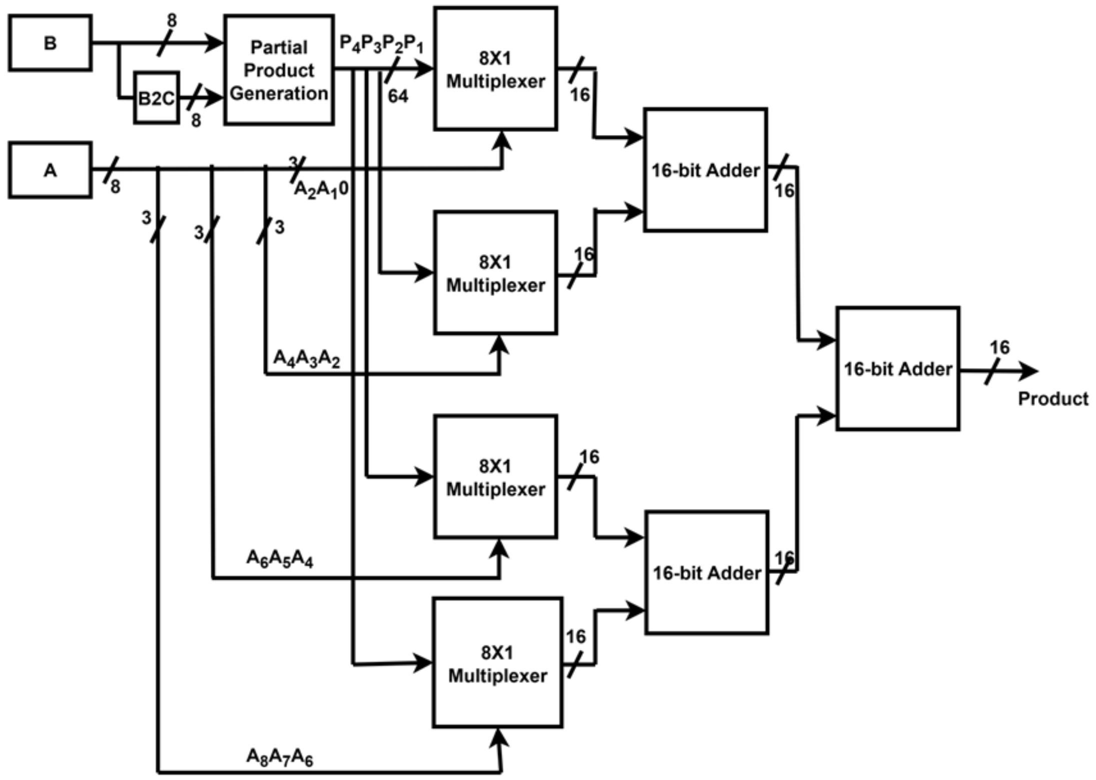


Figure 1 : Proposed Design Booth Encoding Scheme

Overall, the proposed optimization strategy effectively minimizes the hardware area while simultaneously enhancing the performance, making it highly suitable for efficient multiplier designs.

3.2 Partial Product Generation in Proposed Design

Partial product generation is carried out by precomputing the following four expressions:

$P_1 = b$ (4)

$P_2 = \sim b + 1$ (Two's complement of b) (5)

$P_3 = \{b, 1'b0\}$ (Left shift by 1 using bit append method) (6)

$P_4 = \sim(\{b, 1'b0\}) + 1$ (Two's complement of b<<1) (7)

These partial products are computed once and subsequently used during the multiplication process. The selection of the appropriate partial product is governed by the Booth encoding scheme.
To mitigate fan-out issues resulting from multiple loading conditions, buffer stages are inserted as necessary to maintain signal integrity and timing performance.

3.3 Partial Product Selection in Proposed Design

Following the generation of the four partial products, the selection process is carried out using the Booth-encoded bits. Four 8×1 multiplexers (each 16 bits wide) are employed, where the partial products are mapped to the multiplexer inputs as follows:

The multiplexer inputs M_0 to M_7 select the following partial products respectively:
$\mathbf{M_0 \rightarrow 0, M_1 \rightarrow P_1, M_2 \rightarrow P_1, M_3 \rightarrow P_2, M_4 \rightarrow P_3, M_5 \rightarrow P_4, M_6 \rightarrow P_1, M_7 \rightarrow 0}$

The selection lines and corresponding input bits are as follows:

- **S0** $\rightarrow A_2, A_1, A_0$
- **S1** $\rightarrow A_4, A_3, A_2$
- **S2** $\rightarrow A_6, A_5, A_4$
- **S3** $\rightarrow A_8, A_7, A_6$

Each triplet of Booth-encoded bits controls the selection of the corresponding precomputed partial product, ensuring efficient and accurate multiplication.

3.4 Adder

Adders are fundamental components in digital systems, especially within Arithmetic Logic Units (ALUs), where they support a variety of arithmetic and logical operations. Among all arithmetic operations, addition is the most basic and frequently used.

3.4.1 Ripple Carry Adder (RCA):

RCA connects a series of full adders, where the carry propagates sequentially from LSB to MSB. While simple in design, its performance is limited by the cumulative carry propagation delay across stages.

3.4.2 Carry Look-Ahead Adder (CLA):

CLA reduces delay by computing carry signals in parallel using generate and propagate logic. This eliminates the sequential dependency of RCAs and significantly improves addition speed.

3.4.3 Carry Select Adder (CSLA):

CSLA enhances performance by precomputing two possible outcomes for each stage (carry-in = 0 and carry-in = 1) using parallel RCAs. A multiplexer then selects the correct result, reducing critical path delay at the cost of increased hardware.

3.5 FIR Filter Structure

This section presents the simulation results of a 4-tap direct-form FIR (Finite Impulse Response) filter implemented using Radix-4, Radix-8, and Radix-16 Booth multipliers. The filter structure consists of a series of delay elements ("D") that sequentially store input samples, enabling access to current and past inputs. Each delayed input is multiplied by a predefined filter coefficient using the respective radix-based multiplier. The products are then summed using adders to produce the final output.

This weighted sum of current and previous input values provides precise control over the signal's frequency response, making the architecture suitable for high-performance applications such as audio filtering, data smoothing, and communication systems. The use of optimized multiplier designs in the filter significantly impacts power, area, and delay performance across different radix configurations.

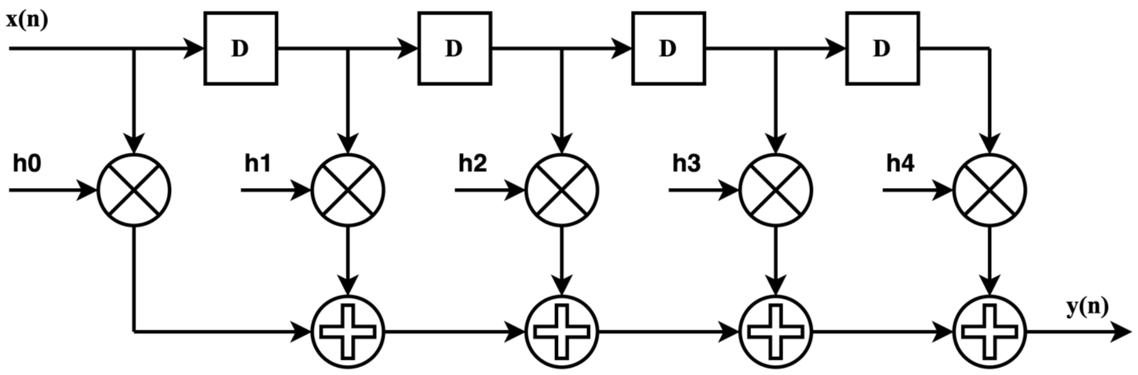


Figure 3 : FIR Filter Structure.

4. Simulation and Result Analysis

This section presents the simulation outcomes and synthesis-based analysis of the proposed adder, multiplier, and FIR filter architectures. The functional verification was carried out using the Xilinx Vivado Design Suite, while detailed hardware analysis was performed using the **Cadence Genus RTL Compiler targeting a 180nm standard cell library**. Key performance metrics such as cell area, power consumption, and delay were extracted from Genus post-synthesis reports to assess the silicon efficiency of each design.
For reference, Vivado analysis was also conducted using the Artix-7 (XC7A35T) FPGA to verify functionality and compare preliminary logic utilization in terms of Look-Up Tables (LUTs), Flip-Flops, DSP slices, and Block RAMs. However, the primary hardware evaluation focuses on the ASIC-style results using 180nm CMOS technology, providing a realistic estimation of the area-power-speed trade-offs for VLSI implementation.

4.1 Simulation and Results of Adder Circuit

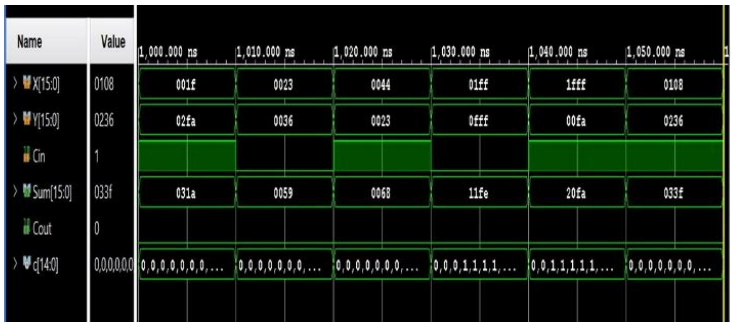
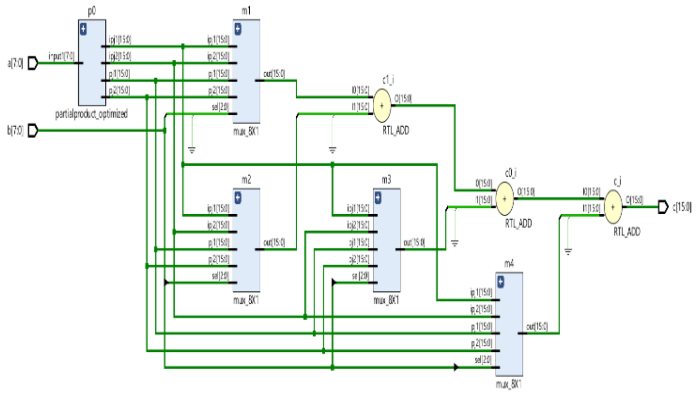

Name
Value
1,000.000 ns
1,010.000 ns
1,020.000 ns
1,030.000 ns
1,040.000 ns
1,050.000 ns
X[15:0] 0108 001f 0023 0044 01ff 1fff 0108
Y[15:0] 0236 02fa 0036 0023 0fff 00fa 0236
Cin 1
Sum[15:0] 033f 031a 0059 0068 11fe 20fa 033f
Cout 0
c[14:0] 0,0,0,0,0,0 0,0,0,0,0,0,0,... 0,0,0,0,0,0,0,... 0,0,0,0,0,0,0,... 0,0,0,1,1,1,1,... 0,0,1,1,1,1,1,... 0,0,0,0,0,0,0,...

Figure 4 : Simulation Result of Ripple Carry Adder.

Table 3 Comparison of Adder Circuit

Parameter	**RCA**	**CLA**	**CSLA**
Delay (ns)	13.343	13.838	10.825
Logic power (W)	0.113	0.119	0.161
Utilization (LUT)	16	16	23

4.2 Simulation and Results of Multiplier Circuit:



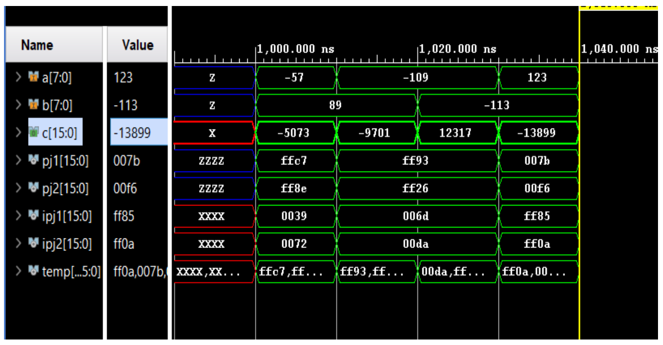

Name
Value
1,000.000 ns
1,020.000 ns
1,040.000 ns
a[7:0] 123 z -57 -109 123
b[7:0] -113 z 89 -113
c[15:0] -13899 x -5073 -9701 12317 -13899
pj1[15:0] 007b zzzz ffc7 ff93 007b
pj2[15:0] 00f6 zzzz ff8e ff26 00f6
ipj1[15:0] ff85 xxxx 0039 006d ff85
ipj2[15:0] ff0a xxxx 0072 00da ff0a
temp[...5:0] ff0a,007b, xxxx,xx... ffc7,ff... ff93,ff... 00da,ff... ff0a,00...

Figure 5 : Simulation of Proposed Multiplier.

Figure 6 : RTL Analysis of Proposed Multiplier Circuit

Table 4 Implemented Comparison of Multiplier

Parameter	**Radix-4[4]**	**Radix-8[5]**	**Radix-16[6]**	**Proposed Design**
Delay (ns)	9.506	16.814	7.711	9.134
Logic power (W)	1.047	1.506	1.321	0.506
Utilization (LUT)	70	144	253	70

Table 5 8-bit Multiplier Implementation result Using Cadence Genus

Parameter	Radix 4			Radix 8			Radix 16		
	Conv. design	Proposed	% Imp.	Conv. design	Proposed	% Imp.	Conv. design	Proposed	% Imp.
Area(μm^2)	4877.95	4143.27	15.06	6016.92	5143.99	14.51	7824.46	8063.45	-2.43
Power(μW)	379.05	366.06	3.43	522.69	373.66	28.52	794.34	519.08	34.65
Delay(ns)	10.18	8.70	14.54	10.33	9.11	11.81	13.26	10.26	22.62

Table 6 64-bit Multiplier Implementation using cadence Genus

Parameter	Radix 4			Radix 8			Radix 16		
	Conv. design	Proposed	% Imp.	Conv. design	Proposed	% Imp.	Conv. design	Proposed	% Imp.
Area(μm^2)	613977.49	253175.55	58.76	405094.28	260313.55	35.74	528143.90	359690.7	31.89
Power(mW)	67.57	80.85	-16.43	89.84	75.16	16.34	113.86	60.76	46.64
Delay (ns)	155.62	84.39	45.77	97.14	85.49	11.99	117.64	92.56	21.32

4.3 Simulation and Results of FIR Filter Structures

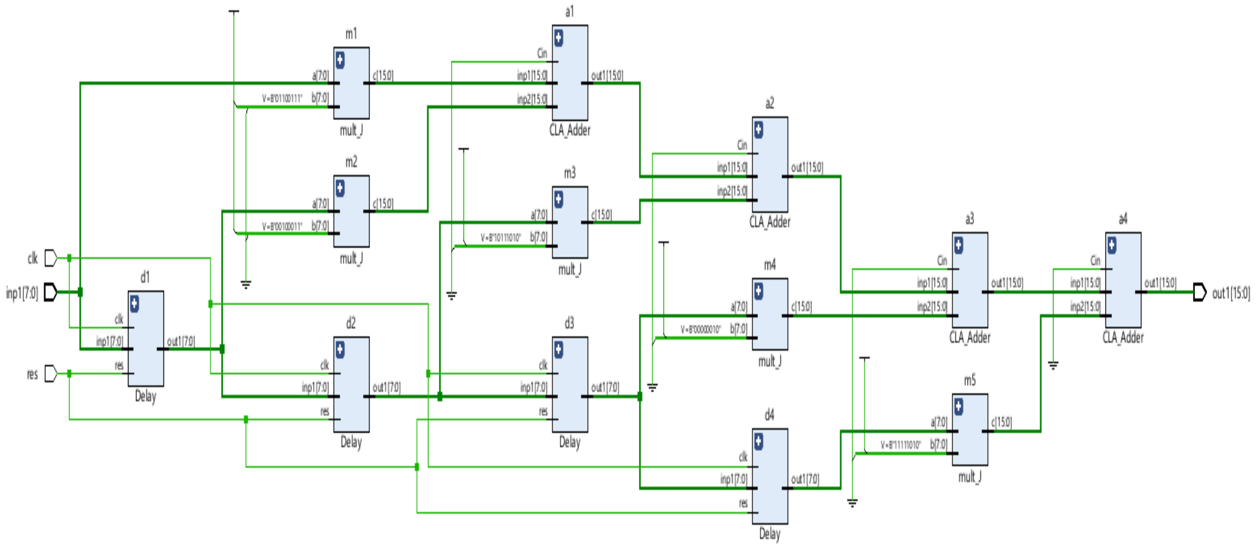


Figure 7 : RTL Schematic of Radix-4 FIR filter Structure.

Table 7 Comparison of FIR Filter

Parameter	**Filter1**	**Filter2**	**Filter3**
Adder	CLA	CLA	CLA
Multiplier	Radix 4	Radix 8	Radix 16
Delay (ns)	22.094	24.918	27.720
Logic power (W)	3.266	5.255	8.847
Utilization (LUT)	297	1088	359

5. Results and Discussion

The proposed CPPG-based Booth multipliers were synthesized using Cadence Genus (180 nm) and compared against conventional designs across Radix-4, Radix-8, and Radix-16 configurations for both 8-bit and 64-bit word lengths.
In 8-bit designs, CPPG integration led to notable improvements:

- Area was reduced by 15.06% (Radix-4) and 14.51% (Radix-8), while Radix-16 saw a minor increase (−2.43%) due to encoding complexity.
- Power consumption improved significantly in Radix-8 (28.52%) and Radix-16 (34.65%), with marginal gain in Radix-4 (3.43%).
- Delay reduced across all configurations, with Radix-16 achieving the best improvement (22.62%), followed by Radix-4 (14.54%) and Radix-8 (11.81%).

For 64-bit multipliers, the proposed CPPG-based designs scaled efficiently:

- Area savings were substantial, especially for Radix-4 (58.76%), followed by Radix-8 (35.74%) and Radix-16 (31.89%).

- Power was reduced most in Radix-16 (46.64%), while Radix-8 showed moderate savings (16.34%). Radix-4 incurred a power penalty (−16.43%) likely due to increased dynamic activity.
- Delay improved significantly in Radix-4 (45.77%), with Radix-16 and Radix-8 achieving 21.32% and 11.99%, respectively.
- Overall, the CPPG methodology offers consistent area and delay benefits across all configurations, with power efficiency improving notably at higher radix levels and larger bit-widths.

6. Future Work

Future enhancements can focus on integrating advanced adder architectures such as Kogge-Stone, Brent-Kung, or Ladner-Fischer to further reduce the critical path in partial product summation. Pipelining strategies may be employed to improve throughput and timing closure, especially in high-frequency designs. For energy-constrained applications, approximate adders can be explored to achieve power and area savings with acceptable accuracy loss.

Further optimization avenues include efficient binary to 2's complement conversion, enhanced signed shift-and-add techniques, and low-overhead encoding schemes for higher-radix multipliers. Additionally, the proposed architecture can be extended to multi-operand multiplication and FIR filter implementations with higher tap counts, where component reuse and architectural-level optimization can offer improved power, performance, and area (PPA) trade-offs.

7. Conclusion

The proposed CPPG-based Booth multipliers demonstrate notable gains in area, delay, and power across Radix-4, 8, and 16 configurations. Radix-4 suits area- and speed-critical designs, Radix-8 offers balanced performance, while Radix-16 is ideal for power-sensitive applications. Overall, CPPG enhances parallelism and scalability, making it well-suited for modern high-performance multipliers.

References:

[1] David Money Harris, Sarah Harris, "Digital Design and Computer Architecture", Morgan Kaufmann publishers
[2] Kundeti Nagarjuna, R. Krishna, "Efficient Implementation of High-Speed Low Power Digital Fir Filter Architecture Design by Radix-4 Multiplier", International Journal of Science, Engineering and Technology
[3] Naga Venkata Vijaya Krishna Boppana, Jeevani Kommareddy, Saiyu Ren, "Low-cost and High-performance 8×8 Booth Multiplier ", Circuits, Systems, and Signal Processing (2019) 38:4357–4368
[4] Ahsan Rafiq, Maksim Jenihhin, "An optimized design of delay-and energy-efficient Booth multiplier" in e-Prime - Advances in Electrical Engineering, Electronics and Energy 9 (2024) 100698, ELSEVIER
[5] Boppana, N.V.V.K., Ren, S.: Simple low power-delay-product parallel signed multiplier design using radix-8 structure with efficient partial product reduction. *J. Eng.* 2023, e12296 (2023). https://doi.org/10.1049/tje2.12296
[6] Serap Ceklia, *, Ali Akmanb," A high speed pipelined radix-16 Booth multiplier architecture for FPGA implementation" *AEUE - International Journal of Electronics and Communications 185 (2024) 155435*
[7] Lad Ketan Khandubhai, "A High-Performance FIR filter using Logarithmic Number System".
[8] Kher Divya, "Comparatively Analysis of Approximate Multiplier for Signal Processing Application"
[9] Gandhi Dhavalkumar Rameshchandra, "Low Power High Performance Multiplier"
[10] References for TWS
https://www.qualcomm.com/products/features/truewireless
https://www.silergy.com/use/True+Wireless+Stereo+%28TWS%29
https://www.embedded.com/case-study-optimizing-ppa-with-risc-v-custom-extensions-in-tws-earbuds/