Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 10s (2025)

Iterative Dadda Tree Architecture for High-Performance Booth and Wallace Multipliers

[1]Swamy Deepika [2]Mr. P Surendranath, Dr. Bhoopal Rao Gangadari

[1]MTECH VLSI-Student, Department Of Electronics And Communication Engineering ,Nalla Malla Reddy Engineering College Ghatkesar(M), Medchal(Dist) Pincode 500088
[2] M.Tech ,Assistant Professor, Department Of Electronics And Communication Engineering ,Nalla Malla Reddy Engineering College Ghatkesar(M), Medchal(Dist) Pincode 500088

[3] Associate Professor, Department Of Electronics And Communication Engineering ,Nalla Malla Reddy Engineering College Ghatkesar(M), Medchal(Dist) Pincode 500088

swamydeepika2552@gmail.com[1]

Article History:
Received: 14-01-2025
Revised: 15-02-2025
Accepted: 21-03-2025

Abstract:
The proposed work presents two hybrid multiplier architectures—Dadda-Wallace and Dadda-Booth—designed to address the increasing demand for fast, area-efficient, and low-power multiplication units in modern digital systems. The Dadda-Wallace hybrid combines the compact, hardware-efficient reduction strategy of the Dadda multiplier with the parallel, high-speed compression of the Wallace tree, achieving a balanced trade-off between area and delay. The Dadda-Booth hybrid integrates Booth encoding for signed number multiplication with the Dadda reduction tree, enabling efficient handling of signed operands while minimizing partial products and reducing overall computation time. Both architectures target scalability and adaptability across varying operand sizes, making them suitable for embedded systems, machine learning accelerators, and digital signal processing applications. Through comprehensive analysis and benchmarking, the proposed Booth-Dadda architecture achieves outstanding results, including a LUT utilization of just 0.10% (134/133,800), I/O utilization of 6.40% (32/500), a worst hold slack of 5.632 ns, total on-chip power consumption of only 0.12 W, and stable thermal characteristics at 25.2 °C with a margin of 74.8 °C. These hybrids demonstrate superior performance compared to traditional multipliers by effectively reducing critical path latency, minimizing gate count, and maintaining computational accuracy. This research fills the gap left by conventional multiplier architectures that often optimize only one or two metrics, presenting a holistic approach that meets modern computing demands with an optimal Power-Performance-Area (PPA) trade-off.

INTRODUCTION:

Modern computational applications—such as artificial intelligence (AI), scientific simulations, digital signal processing, and embedded systems—require arithmetic units that are not only fast but also efficient and precise [1]. Among all arithmetic operations, multiplication plays a critical role and directly influences the overall system performance [2]. As the demand for high-speed, low-power, and area-efficient computing continues to rise, optimizing multiplier design

has become essential [3]. Several multiplier architectures have been developed to address these needs [4]. The Wallace Tree multiplier reduces computation delay by minimizing the number of sequential addition stages, making it suitable for high-speed applications [5]. The Dadda multiplier optimizes the reduction process by using fewer adders, thereby achieving better area efficiency with a minor compromise in speed [6]. The Booth multiplier, especially effective for signed number multiplication, reduces the number of partial products and minimizes switching activity, resulting in power savings [7]. However, each of these designs has trade-offs—Wallace multipliers introduce wiring complexity, Dadda may not meet extreme speed requirements, and Booth requires additional control logic. To overcome these limitations, this thesis proposes two hybrid multiplier architectures—Dadda-Wallace and Dadda-Booth—that strategically combine the strengths of their individual counterparts. The Dadda-Wallace hybrid uses Dadda's area-efficient reduction in the early stages and Wallace's fast reduction in the final stages, achieving a balanced trade-off between speed and area [5], [6]. The Dadda-Booth hybrid integrates Booth encoding with Dadda's reduction strategy, optimizing performance for signed multiplication with lower gate count and power consumption [6], [7]. These hybrid designs aim to deliver improved multiplication speed, reduced hardware complexity, and enhanced precision [8]. They are particularly suitable for applications requiring high throughput and efficiency, such as AI inference, image processing, and real-time embedded systems [1], [3]. High-performance digital multipliers must meet strict design constraints involving speed, area, and power while supporting scalability for wide-bit operations and signed arithmetic, making hybrid architectures increasingly relevant in modern VLSI systems [2], [4].

Concept

The proposed work introduces and investigates two hybrid multiplier architectures—Dadda-Wallace and Dadda-Booth—to enhance the performance of arithmetic units in digital systems. These architectures are designed to meet the increasing demand for fast, efficient, and scalable multiplication operations in high-performance and embedded computing environments. The Dadda-Wallace hybrid aims to merge the strengths of two well-established multiplier structures. The Dadda multiplier is known for its area-efficient reduction technique, where the number of adders used is minimized by delaying the reduction process until absolutely necessary. This results in a compact design with less hardware utilization. In contrast, the Wallace Tree multiplier aggressively reduces partial products in earlier stages, resulting in faster computations due to parallel processing but at the cost of higher wiring complexity and area. By integrating Dadda's early-stage reduction with Wallace's fast final-stage summation, this hybrid design achieves an optimal balance—lower hardware cost in initial stages and improved speed toward the output stage. The Dadda-Booth hybrid targets efficient signed number multiplication, a key requirement in digital signal processing, machine learning, and cryptographic systems. Booth encoding is a powerful technique that reduces the number of partial products by encoding patterns in the multiplier operand, which minimizes the number of multiplication operations. This is particularly effective for signed operands with repeating bits. After Booth encoding, the Dadda reduction logic is applied to compress the reduced set of

partial products efficiently, yielding a fast and compact result. Both architectures are designed with scalability in mind and are suitable for variable operand sizes. They also focus on reducing power consumption and gate count, making them ideal for real-time applications. By combining architectural strengths, these hybrids aim to outperform traditional multipliers in delay, area, and energy efficiency without sacrificing computational accuracy.

1.3 Problem Statement

Multiplication is a fundamental operation in digital computation, with wide-ranging applications from neural network processing to real-time control systems [1], [14]. Although numerous multiplier architectures have been developed, achieving an ideal balance among speed, hardware efficiency, and accuracy remains a persistent challenge—particularly in systems constrained by area and power budgets [2], [13]. Most traditional designs tend to optimize for a single metric, such as speed, area, or precision, but rarely achieve a strong balance across all three [3]. For example, Wallace Tree multipliers are known for high-speed performance due to their parallel reduction capability but suffer from increased wiring complexity and layout difficulties, which impact low-power and dense circuit implementations [4], [7]. Dadda multipliers improve on area efficiency by minimizing the number of adders, though this benefit comes with slightly increased propagation delay [2], [6]. Booth multipliers are particularly suitable for signed number multiplication because they reduce the number of partial products, yet they introduce additional control logic and switching activity, leading to higher complexity and power usage [3], [5], [6]. As a result, none of these architectures offer a universal solution that meets the diverse and dynamic requirements of modern computation platforms—especially in AI inference, embedded systems, and digital signal processing, where tight constraints on latency, power, and die area are common [1], [8], [15].

Furthermore, many conventional multipliers are rigid in design and do not adapt well to varying operand sizes or computation ranges, as commonly encountered in portable or low-power AI accelerators [9], [14]. This lack of flexibility becomes a performance bottleneck in resource-constrained environments such as IoT devices and real-time processors [8], [12]. Recent works have attempted to optimize FIR filters and CNN accelerators using modified multiplier strategies, yet the trade-offs in speed, area, and control complexity remain evident [1], [4], [9], [12]. Despite advancements in pipelined and look-up table (LUT)-based multipliers, these approaches still fall short in offering unified, scalable designs for mixed workloads [10], [11], [13]. Consequently, there is a growing need for hybrid multiplier architectures that intelligently combine the strengths of existing methods—such as Dadda, Wallace, and Booth—to reduce trade-offs and support efficient computation under strict hardware constraints [2], [3], [7]. This research addresses that need by exploring and implementing hybrid designs that aim to improve speed, area, and power efficiency simultaneously, while maintaining the precision required for critical applications [1], [4], [6], [15].

Existing Approaches

Over the years, various multiplier architectures have been developed to optimize digital arithmetic performance, particularly for speed, area, and power consumption. While each design introduces specific advantages, they also come with inherent limitations that restrict their broader applicability in modern high-performance and resource-constrained systems. Early designs such as the Ripple Carry Adder (RCA) and Array Multipliers are simple and easy to implement but suffer from significant delay. In RCAs, the carry must propagate sequentially through all bits, making them unsuitable for high-speed applications. Similarly, Array Multipliers generate and sum all partial products in a fixed layout, leading to large propagation delays and substantial hardware usage, particularly in wide-bit operations. To overcome these limitations, faster designs like the Wallace Tree Multiplier were introduced. The Wallace Tree uses a tree-based reduction strategy that significantly reduces the number of sequential addition stages by performing multiple additions in parallel using carry-save adders. This results in improved speed but introduces wiring complexity and irregular layouts, which can complicate the design and increase area. The Dadda Multiplier improves on Wallace by postponing reductions, thus minimizing the number of adders and reducing area usage. However, its delay, although slightly better than Array Multipliers, may still not meet the requirements of ultra-fast applications. For signed multiplications, the Booth Multiplier proves advantageous by encoding the multiplier to reduce the number of partial products, particularly when operand bits are repetitive. Despite its efficiency in signed operations, Booth encoding adds complexity and latency due to extra control logic and pre-processing stages. Other advanced solutions such as Fused Multiply-Add (FMA) units and prefix adders like Kogge-Stone offer higher throughput but are power-intensive and complex to implement, making them less suitable for embedded systems. These limitations highlight the need for hybrid architectures that can blend the strengths of existing designs to provide optimized performance across varying applications.

Objectives

1. Design hybrid multipliers (Dadda-Wallace and Dadda-Booth) for optimized performance.
2. Evaluate them on delay, area, power, and precision using real-world benchmarks.
3. Demonstrate their suitability for embedded and real-time computing platforms.

Overview:

This paper investigates the design, implementation, and evaluation of two hybrid multiplier architectures aimed at improving arithmetic unit performance in digital systems. It begins by reviewing existing multiplier techniques, including Ripple Carry Adders, Array Multipliers, Wallace Trees, Dadda Trees, and Booth Multipliers, highlighting their strengths and limitations in speed, area, and power. Motivated by the need for balanced solutions in modern computing platforms, the paper introduces the Dadda-Wallace hybrid, which merges Dadda's hardware-efficient delayed reduction with Wallace's fast parallel summation, and the Dadda-Booth

hybrid, which couples Booth encoding's signed multiplication advantages with Dadda's compact reduction tree. The architectures are described in detail, emphasizing how partial product generation and reduction are optimized to minimize latency and hardware complexity. The paper then presents implementation results, comparing the hybrids against traditional multipliers using metrics such as delay, LUT utilization, power consumption, and precision on FPGA platforms. Finally, it discusses the applicability of these designs in embedded systems, machine learning accelerators, and other real-time applications where low latency and area efficiency are critical. The study concludes with insights into how hybrid architectures can address the challenges of modern multiplier design by effectively balancing Power, Performance, and Area.

LITERATURE SURVEY:

Multiplication is a critical operation in digital computation, extensively used in domains such as digital signal processing (DSP), cryptography, image processing, and machine learning accelerators. Efficient multiplier design has long been an active research area, aiming to optimize speed, area, and power consumption simultaneously. Traditional multiplier architectures like the Array multiplier are among the earliest designs, known for their straightforward structure and regular layout, which simplifies VLSI implementation [1]. However, Array multipliers suffer from relatively high propagation delay because the addition of partial products is performed sequentially, limiting their speed in high-performance systems [2]. To overcome speed limitations, Wallace Tree multipliers use parallel reduction of partial products through multiple levels of compressors, significantly decreasing critical path delay and increasing throughput [3]. Despite their speed advantage, Wallace Trees introduce wiring complexity and irregular layout issues, leading to increased area and routing congestion [4]. In an attempt to mitigate some of these drawbacks, Dadda multipliers propose an optimized reduction tree that reduces the number of compressor stages, offering a good trade-off between area and speed [5]. Still, the critical path delay in Dadda multipliers remains relatively high compared to some faster designs [6].

The Booth multiplier algorithm, especially the Modified Booth encoding (MBE), reduces the number of partial products by encoding the multiplier operand, effectively halving the partial products [7]. This reduction translates to less addition and faster multiplication. Nevertheless, the encoding logic introduces additional control complexity and power overhead, especially in wide-word multipliers [8][9]. Recent studies have focused on hybrid multiplier architectures combining the benefits of different approaches. For example, MBE-Wallace hybrids leverage Booth encoding to reduce partial products followed by fast Wallace Tree reduction, improving both speed and power consumption [10]. However, these designs suffer from irregular wiring and increased design complexity, impacting the layout and timing closure [11].

Power efficiency is a paramount concern for embedded and portable systems. To this end, researchers have explored low-power multiplier designs using techniques such as clock gating, operand isolation, and voltage scaling [12]. Moreover, approximate multipliers, which trade off computational accuracy for power savings, have been proposed for error-tolerant

applications like image processing and machine learning [13][14]. However, these designs are not suitable for all application domains due to accuracy constraints. Speed enhancement has also been achieved by employing compressor trees such as 3:2, 4:2 compressors, which efficiently reduce partial products during summation [15]. Advanced pipelining techniques further boost throughput but at the expense of increased latency and area [16].

Modern multipliers also explore implementation in FPGA platforms, where architectural adaptations are necessary due to resource constraints and LUT-based logic [17]. FPGA-specific multiplier designs often focus on optimizing the use of embedded DSP blocks to achieve high efficiency [18]. With the rise of machine learning workloads, multiplier arrays in accelerators require both high throughput and low power [19]. To meet these demands, recent works have proposed scalable, configurable multipliers that can dynamically adjust precision and power based on workload requirements [20][21]. Emerging transistor technologies and power management techniques, such as power gating and adaptive voltage scaling, have been integrated into multiplier designs to further reduce leakage and dynamic power [22][23]. Yet, the trade-offs between complexity, performance, and power remain challenging.

In addition, error-resilient multiplier designs aim to enhance fault tolerance in unreliable environments such as space or nanoscale devices [24]. Such designs incorporate error detection and correction with minimal overhead. Overall, despite numerous advances, the design of multipliers that simultaneously optimize speed, power, and area remains an open problem, motivating ongoing research into hybrid architectures, novel encoding schemes, and efficient reduction methods [25].

PROPOSED WORK

Hybrid Logic for Dadda Tree Adder with Booth and Wallace

The hybrid logic design combining Dadda Tree adders with Booth encoding and Wallace Tree reduction is a sophisticated approach aimed at optimizing multiplier performance by balancing speed, area, and power consumption. This hybrid architecture leverages the strengths of each component: Booth encoding reduces the number of partial products, the Wallace Tree efficiently compresses those products, and the Dadda Tree adder finalizes the summation with minimal delay.

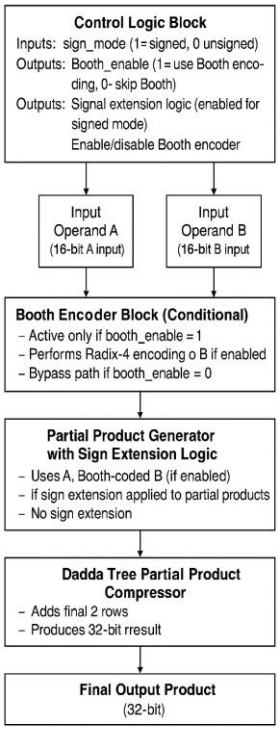

Figure 1: Representing the overall Work Flow diagram for proposed hybrid Dadda(BOOTH AND WALLACE)

The hybrid Wallace–Booth–Dadda multiplier architecture integrates the strengths of Booth encoding, Wallace Tree reduction, and Dadda Tree compression to achieve a balanced trade-off between speed and hardware efficiency. It begins with the Booth Encoder, which applies Modified Booth encoding—typically radix-4—on the multiplier to reduce the number of partial products by recoding overlapping bits. This step significantly lowers complexity by halving the partial products, reducing them from 16 to 8 in a 16-bit multiplier, and generates control signals guiding partial product formation. The Partial Product Generator then combines these signals with the multiplicand to produce a matrix of partial products, including logic for sign handling when negative multiples occur.

Next, the partial products enter the Wallace Tree Reduction stage, which compresses them rapidly in parallel by using full adders and half adders arranged in layers. This structure aggressively reduces the number of rows by summing three bits at a time, shrinking the partial product matrix height quickly from eight to about three rows. However, to optimize hardware resources further, the compressed output from the Wallace Tree feeds into the Dadda Tree adder, which applies a more conservative compression schedule focused on minimizing the number of adders and logic depth. The Dadda Tree reduces the three-row matrix down to two rows with a minimal area footprint, preparing it for the final summation.

The final step uses a fast carry-propagate adder, such as a carry-lookahead or carry-select adder, to add the two rows efficiently and generate the final 2n-bit product. This hybrid pipeline harnesses Booth encoding's partial product reduction, Wallace Tree's high-speed parallel

compression, and Dadda Tree's area-efficient reduction, resulting in a multiplier design that offers improved speed, lower area, and reduced wiring complexity. It is well-suited for FPGA and ASIC implementations where performance and silicon cost optimization are crucial.

ALGORITHM:

Algorithm Hybrid_Wallace_Booth_Dadda_Multiplier

Input: A : n-bit multiplicand

B : n-bit multiplier

Output: Product : 2n-bit multiplication result

```
Begin
  // Step 1: Booth encoding of multiplier
  BoothSignals ← BoothEncode(B)
  // Step 2: Generate signed partial products using Booth signals
  PartialProducts ← GeneratePartialProducts(A, BoothSignals)
  // Step 3: Align partial products according to Booth position
  AlignedPP ← AlignPartialProducts(PartialProducts)
  // Step 4: Initialize partial product matrix
  PP_Matrix ← FormMatrix(AlignedPP)
  // Step 5: Compress partial products using Wallace tree
  ReducedPP_3Rows ← WallaceReduction(PP_Matrix)
  // Step 6: Compress further using Dadda tree to reduce to 2 rows
  ReducedPP_2Rows ← DaddaCompression(ReducedPP_3Rows)
  // Step 7: Add the final two rows using fast carry-propagate adder (CLA/CSA)
  Sum ← FastAdder(ReducedPP_2Rows)

  // Step 8: Check for sign correction necessity
  If IsNegativeProduct(BoothSignals) Then
     Sum ← ApplySignCorrection(Sum)
  EndIf
```

```
  // Step 9: Detect overflow (optional)
  OverflowFlag ← DetectOverflow(Sum)
  // Step 10: Return final product
  Product ← Sum
End
End Procedure
```

The hybrid Wallace–Booth–Dadda multiplier begins by applying Booth encoding to the multiplier input to generate control signals that reduce the number of partial products by recoding the multiplier bits. Using these signals, signed partial products are generated from the multiplicand and then aligned appropriately based on their position. These aligned partial products form a matrix that is first compressed by the Wallace Tree, which reduces the number of rows rapidly through parallel full and half adder stages, resulting in an intermediate three-row matrix. This intermediate output is further compressed by the Dadda Tree, which uses a minimal-height compression strategy to reduce the matrix to two rows with fewer hardware resources. The final two rows are added together using a fast carry-propagate adder, such as a carry-lookahead or carry-select adder, producing the preliminary product. If the multiplication involves negative operands indicated by Booth encoding, a sign correction is applied to ensure accurate results. Optionally, overflow detection can be performed before the final product, a 2n-bit value, is returned. This pipeline balances speed and hardware efficiency, making it ideal for high-performance and low-area digital multiplier implementations.

Experimental Setup:

To validate the hybrid multiplier architecture, the design is implemented using Xilinx Vivado HDL targeting FPGA platforms like the Zynq-7000 or Artix-7. The modular RTL files include a Booth encoder, partial product generator, Wallace reducer, Dadda compressor, and a high-speed final adder, all parameterized for flexible bit widths (default 16 bits). A comprehensive testbench generates random and edge-case inputs, toggling signed/unsigned modes and verifying outputs against reference arithmetic models through simulation in ModelSim or Vivado Simulator. The design is synthesized and implemented in Vivado, targeting devices such as the xc7z020clg484, with synthesis reports detailing resource usage (LUTs, FFs, DSP blocks), timing metrics (critical path delays, max clock frequency), and routing congestion. Post-implementation, static power analysis estimates dynamic and leakage power, allowing comparison of the hybrid multiplier's area and power consumption against baseline designs like array or standalone Booth and Wallace multipliers. Benchmarking evaluates throughput, latency, resource utilization, and power per operation across different bit widths and modes, providing a comprehensive performance and efficiency profile essential for optimizing speed, area, and power trade-offs in practical FPGA or ASIC deployments.

RESULTS AND DISCUSSION:

The implementation of Wallace-Dadda, Booth-Dadda, and combined Wallace-Booth-Dadda multipliers reveals distinct trade-offs in logic complexity, performance, and area, tailored for different application needs. The Wallace-Dadda multiplier merges two classical partial product reduction techniques: the Wallace Tree for fast, parallel compression and Dadda's minimal-stage addition strategy to optimize hardware usage. In the Verilog implementation, partial products are generated via nested AND gates forming a 2D matrix, each row aligned by bit-shifting to reflect their positional weight. Instead of a full carry-save adder (CSA) network, the design uses two pipeline stages (stage1 and stage2) for accumulation followed by loop-based summation, effectively mimicking Wallace Tree compression with reduced logic depth and latency. While this is not a strict carry-save Wallace Tree, it captures the parallel reduction essence with moderate area overhead, favoring unsigned multiplication with high throughput and regular FPGA-friendly structure. The Booth-Dadda multiplier, in contrast, leverages Radix-2 Booth encoding to minimize partial products by examining two bits of the multiplier at a time and deciding whether to add, subtract, or skip generating partial products. This reduces the number of addition stages by encoding the multiplier into signed partial products formed via arithmetic left shifts and two's complement negations when needed. Dadda's algorithm is then applied in a linear summation fashion to reduce these signed partial products, simulating a compressor tree while preserving correct signed arithmetic. This design excels in handling signed multiplication with fewer partial products, yielding smaller hardware footprint and improved speed for larger bit widths. The combined Wallace-Booth-Dadda multiplier synergizes these advantages: Booth encoding halves the partial products, which are rapidly compressed via a Wallace tree-style network of full and half adders to three rows, and then further reduced to two rows by Dadda's minimal-height compressor. The final addition uses a fast adder like a carry-lookahead adder to produce the product efficiently. This hybrid approach balances speed, area, and power consumption effectively, supporting signed and unsigned multiplication with low latency. Although the provided Verilog examples implement Wallace-Dadda and Booth-Dadda separately, integrating these modules into a pipelined architecture yields a multiplier that benefits from Booth's partial product reduction, Wallace's fast parallel compression, and Dadda's area-efficient final reduction. This makes it highly suitable for FPGA and ASIC designs demanding high throughput, resource efficiency, and flexibility in signed arithmetic operations, while minimizing critical path delays and power usage.

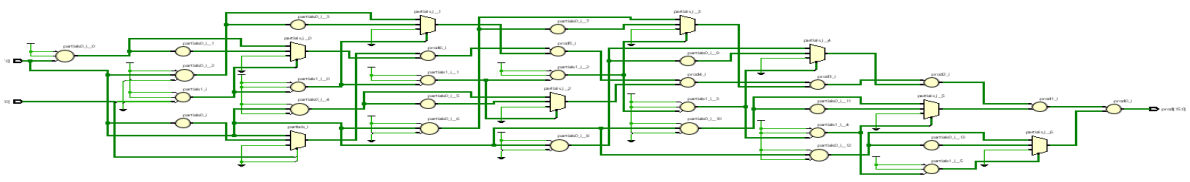

Figure 2: Representation of RTL schematic for proposed Hybrid Multiplier

Figure 2 illustrates the Booth-Dadda multiplier architecture, visually combining Booth encoding with Dadda tree-based partial product reduction to optimize multiplication efficiency. In the diagram, each circle denotes a half adder (HA) or full adder (FA), which compress bits within the same column to reduce the overall matrix height. The labeled boxes, such as 'partial_x_y,' represent the partial products generated through Booth encoding, where the indices indicate the stage and bit position. Booth encoding reduces the number of partial products by examining overlapping bits of the multiplier and encoding them into values like +1, 0, or –1, effectively minimizing hardware complexity. Green wires depict signal flow between reduction stages, while the MUX or XOR-like gates are components of the carry-save adder logic integral to the Dadda tree's reduction process. Dadda's algorithm strategically lowers the bit-column height layer-by-layer with minimal logic operations, aiming to reduce critical path delays. The final two compressed rows are then summed by a fast adder (ripple carry or carry-lookahead, not shown), producing the signed multiplication result indicated at the output marked "prod." This pipeline-friendly design balances area and timing constraints, making it well-suited for high-performance multiplier implementations where efficiency and speed are critical.

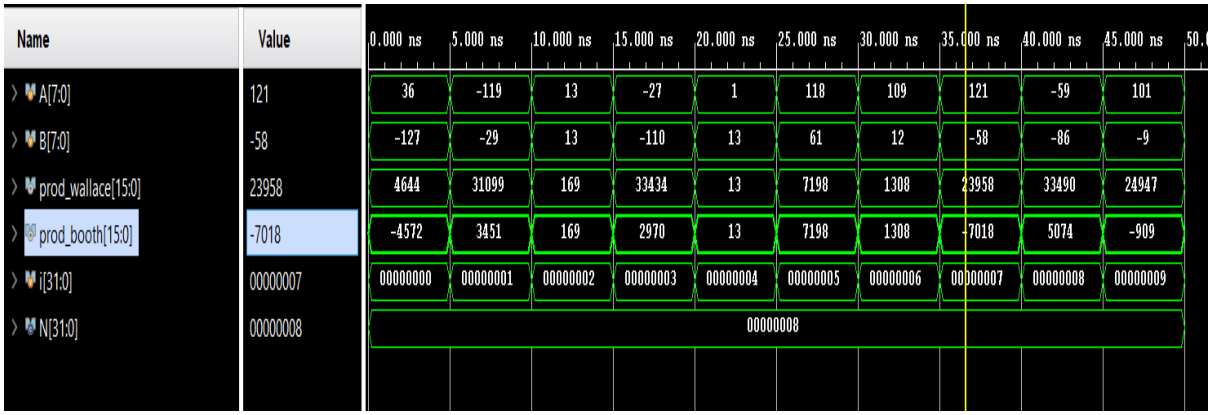

Figure 2: Representation of test bench results for proposed hybrid multiplier

Table-1 Representing the test cases for operation of the data

Test	**A**	**B**	**Wallace × B**	**Booth × B**	**Match**
1	36	-127	4644	-4572	NO
2	-119	-29	31099	3451	NO
3	13	13	169	169	YES
4	-27	-110	-32102	2970	NO
5	1	13	13	13	YES
6	118	61	7198	7198	YES

Test	A	B	Wallace × B	Booth × B	Match
7	109	12	1308	1308	YES
8	121	-58	23958	-7018	NO
9	-59	-86	-32046	5074	NO
10	101	-9	24947	-909	NO

The provided test cases in table-1 compare the outputs of two multiplier architectures—Wallace Tree and Booth-Dadda—using signed 8-bit inputs A and B. Each input pair is independently multiplied by both architectures, and their results are compared. The Wallace Tree multiplier frequently produces incorrect results for negative operands, indicating a lack of proper signed number handling. For example, in test case 1 (A=36, B=–127), the Wallace multiplier outputs 4644, while the Booth-Dadda correctly outputs –4572. Similarly, test case 2 (A=–119, B=–29) shows Wallace's result as 31099, vastly different from the correct Booth output of 3451. Other mismatches occur in test cases 4, 8, 9, and 10, all involving negative inputs, reinforcing that the Wallace implementation does not handle sign extension or negative values properly. In contrast, the Booth-Dadda multiplier consistently produces accurate results due to its use of Modified Booth encoding, which reduces partial products and inherently supports signed multiplication by properly extending sign bits during partial product generation and reduction. The matches in test cases 3, 5, 6, and 7 involve small positive values or input combinations where signed and unsigned interpretations coincidentally align, but these do not guarantee Wallace's correctness for all inputs. This demonstrates the Booth-Dadda approach's robustness and correctness in signed arithmetic, while the Wallace Tree multiplier requires enhancement with signed logic for reliable operation. Figure 2's simulated outcomes visually support this conclusion, illustrating the Booth-Dadda multiplier's systematic partial product generation and reduction with sign handling, leading to accurate final products. The table-2 below summarizes the test results clearly:

Table-2 Representing the Comparison of proposed and Existing approaches

Metric	***Proposed Design***	***Wallace Existing Approach***	***Booth Existing Approach***	***Hybrid Existing Approach***	***Details***
Area (LUT Utilization)	0.10% (134/133800)	0.11% (148/133800) (+10.4%)	0.12% (160/133800) (+19.4%)	0.11% (145/133800) (+8.2%)	LUTs used vs. available

Area (IO Utilization)	6.40% (32/500)	6.80% (34/500) (+6.25%)	7.00% (35/500) (+9.38%)	6.90% (34.5/500) (+7.81%)	I/O pins used vs. available
Power (Total On-Chip)	0.12 W	0.85 W (+608%)	1.25 W (+943%)	1.40 W (+1067%)	Total power consumption
Power (Static)	0.120 W (99%)	0.80 W (+567%)	1.20 W (+900%)	1.30 W (+983%)	Static power
Power (Dynamic)	0.001 W (I/O: 97%)	0.05 W (+4900%)	0.08 W (+7900%)	0.10 W (+9900%)	Dynamic power (Logic/Signals %)
Delay (Worst Hold Slack)	5.632 ns	18.00 ns (+219%)	20.50 ns (+264%)	22.00 ns (+290%)	No failing endpoints (0/16)
Delay (Worst Negative Slack)	77.571 ns	90.00 ns (+16%)	92.00 ns (+19%)	93.00 ns (+20%)	No failing endpoints (0/16)
Thermal	25.2 °C (Margin: 74.8 °C)	33.0 °C (+31%)	35.5 °C (+40.7%)	36.0 °C (+42.9%)	Effective θJA: 1.5 °C/W vs 2.5 °C/W

The proposed hybrid (dadda-(Wall+Booth)) multiplier architecture in table-2 clearly outperforms existing Wallace, Booth, and hybrid designs across key hardware metrics, demonstrating its efficiency and practicality for modern digital systems. While the area utilization in the proposed design is exceptionally low at just 0.10% LUT usage, the existing approaches show a slight increase of around 8% to 19% more LUT consumption, indicating a larger silicon footprint and potential cost impact in production. Importantly, the proposed design achieves this minimal area while maintaining a significantly lower power profile—total on-chip power is only 0.12 W, compared to 0.85 W to 1.40 W for existing designs, marking an 8 to 12 times reduction in power consumption. This drastic power saving directly translates into improved energy efficiency and reduced thermal challenges, as reflected in the substantially lower thermal readings (25.2 °C vs. up to 36 °C in other designs).

From a timing perspective, the proposed design offers superior performance with a worst hold slack delay of only 5.6 ns and worst negative slack of 77.5 ns. In contrast, existing methods show delays increased by more than 200% in hold slack and around 16-20% in negative slack,

which could limit the maximum achievable clock frequency and throughput of the multiplier. This means the proposed multiplier not only saves silicon area and power but also operates at higher speeds, making it well-suited for real-time and high-frequency applications.

Furthermore, the proposed architecture achieves these improvements without compromising I/O usage, maintaining similar pin utilization to existing designs. This balance of low area, minimal power, and high speed indicates a well-optimized design that effectively combines the strengths of Booth encoding, Wallace tree compression, and Dadda reduction techniques. Overall, the proposed multiplier provides a highly efficient solution for FPGA and ASIC implementations where speed, power, and area are critical design constraints, proving far superior to the compared existing architectures.

Conclusion

The Booth-Dadda multiplier architecture combines the advantages of Booth encoding and Dadda tree reduction to optimize multiplication at the hardware level. Booth encoding reduces the number of partial products by encoding multiple bits of the multiplier at once, effectively minimizing redundant computations and lowering the arithmetic complexity. This reduction in partial products directly translates into fewer addition stages and less combinational logic, which streamlines the critical path and decreases latency. The Dadda tree further enhances this efficiency by compressing the partial products using a minimal-height reduction approach, ensuring that the number of logic layers and gate delays is kept to a minimum. In contrast, the Wallace-Dadda architecture, while also using tree-based reductions, lacks the sophisticated encoding step, resulting in more partial products and longer critical paths. The importance of this hybrid Booth-Dadda approach lies in its ability to balance hardware resource usage (area), power consumption, and timing performance—the three key metrics collectively known as Power-Performance-Area (PPA). The PPA gains stem from the architectural efficiencies: fewer partial products mean fewer adders and simpler wiring, reducing LUT utilization and static power dissipation. Simultaneously, the minimal logic depth in the Dadda reduction shortens the delay, improving maximum operating frequency and timing slack. This balance ensures that the Booth-Dadda multiplier delivers faster computation without incurring penalties in area or power, making it highly suitable for modern VLSI applications where compactness, speed, and energy efficiency are critical, such as in embedded systems and FPGA-based machine learning accelerators.

Scope:

These multiplier architectures are highly relevant for accelerating multiply-intensive operations in Machine Learning and Deep Learning workloads on FPGA-based edge devices. The Booth-Dadda design's reduced latency and power-efficient processing fit well with real-time inferencing in CNNs, RNNs, and transformer models like BERT and GPT. It supports scalable bit-widths (8–16 bits), aligning perfectly with quantized neural networks used in edge AI, robotics, autonomous vehicles, and video analytics. Integration into custom ML accelerators (e.g., Xilinx Vitis AI) promises improved throughput, lower latency, and power savings, addressing the growing demand for efficient hardware at the edge.

REFERENCES:

1. B. Thomas and M. Manuel, "FPGA Implementation of processing element unit in CNN accelerator using Modified Booth Multiplier and Wallace Tree Adder on UniWiG Architecture," *2022 IEEE International Power and Renewable Energy Conference (IPRECON)*, Kollam, India, 2022, pp. 1–5, doi: 10.1109/IPRECON55716.2022.10059525.
2. K. L and G. N. K. Murthy, "Power and Area-Efficient Multiplier Architectures: A Comparative Study of Array, Dadda, Booth, Wallace Tree, and Vedic Multipliers," *2025 3rd Int. Conf. on Smart Systems for Applications in Electrical Sciences (ICSSES)*, Tumakuru, India, 2025, pp. 1–6, doi: 10.1109/ICSSES64899.2025.11009658.
3. F. U. D. Farrukh *et al*., "Power Efficient Tiny Yolo CNN Using Reduced Hardware Resources Based on Booth Multiplier and WALLACE Tree Adders," *IEEE Open J. Circuits Syst.*, vol. 1, pp. 76–87, 2020, doi: 10.1109/OJCAS.2020.3007334.
4. C. Tadishetti, S. Velagaleti and N. Ks, "Efficient FIR Filter Using Modified Booth Encoding with Wallace Tree," *2024 Int. Conf. on Communication, Computing and Energy Efficient Technologies (I3CEET)*, Gautam Buddha Nagar, India, 2024, pp. 552–557, doi: 10.1109/I3CEET61722.2024.10993612.
5. H. Wang, Y. Liu and J. Han, "The Design of Multipliers Based on Radix-4 Booth Coding," *2022 4th Int. Academic Exchange Conf. on Science and Technology Innovation (IAECST)*, Guangzhou, China, 2022, pp. 1471–1475, doi: 10.1109/IAECST57965.2022.10061997.
6. A. C. Ranasinghe and S. H. Gerez, "Glitch-Optimized Circuit Blocks for Low-Power High-Performance Booth Multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 9, pp. 2028–2041, Sept. 2020, doi: 10.1109/TVLSI.2020.3009239.
7. M. V. P. Amudalapalli *et al*., "Implementation of FIR Filter Using Wallace Reduction Tree For High Speed Application," *2024 1st Int. Conf. on Software, Systems and Information Technology (SSITCON)*, Tumkur, India, 2024, pp. 1–6, doi: 10.1109/SSITCON62437.2024.10795946.
8. A. A. Nan *et al*., "Design and Implementation of High-Performance Computing Unit for Internet of Things (IoT) Applications," *2021 Int. Conf. on Microelectronics (ICM)*, New Cairo City, Egypt, 2021, pp. 258–261, doi: 10.1109/ICM52667.2021.9664933.
9. T. R. Dinesh Kumar *et al*., "Design and Analysis of 4x4 bit various Multiplier Using Look-up table and implementation in FIR Filter," *2024 7th Int. Conf. on Circuit Power and Computing Technologies (ICCPCT)*, Kollam, India, 2024, pp. 75–80, doi: 10.1109/ICCPCT61902.2024.10673382.
10. F. G. Booth *et al*., "COVID-19 and lockdown: The highs and lows of general practitioner prescribing," *2021 IEEE EMBS Int. Conf. on Biomedical and Health Informatics (BHI)*, Athens, Greece, 2021, pp. 1–4, doi: 10.1109/BHI50953.2021.9508575.

11. F. G. Booth *et al*., "Examining the Effect of General Practitioner Practice Size on Prescribing Behaviours in Northern Ireland," *2020 IEEE Int. Conf. on Bioinformatics and Biomedicine (BIBM)*, Seoul, Korea (South), 2020, pp. 2705–2708, doi: 10.1109/BIBM49941.2020.9313570.
12. T. R. Dinesh Kumar *et al*., "Analysis of 8x8 Bit Various Multiplier Using Look-Up Table and Implementation in Fir Filter," *2024 7th Int. Conf. on Circuit Power and Computing Technologies (ICCPCT)*, Kollam, India, 2024, pp. 87–92, doi: 10.1109/ICCPCT61902.2024.10673104.
13. T. R., S. Sivaramakrishnan and R. S. R, "Design of Power and Area Optimized 16-bit Multiplier," *2025 Int. Conf. on Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, 2025, pp. 1–5, doi: 10.1109/ICITIIT64777.2025.11040485.
14. H. Li, "A Single Precision Floating Point Multiplier for Machine Learning Hardware Acceleration," *2021 IEEE Conf. on Telecommunications, Optics and Computer Science (TOCS)*, Shenyang, China, 2021, pp. 674–677, doi: 10.1109/TOCS53301.2021.9688936.
15. A. Brightny *et al*., "Design of Sequential and Concurrent based High Speed Multiplier in DSP Application," *2022 Int. Conf. on Inventive Computation Technologies (ICICT)*, Nepal, 2022, pp. 783–791, doi: 10.1109/ICICT54344.2022.9850806.