Accelerating Radix-4 Multiplication Using FPGA

Radu R Prasad and Dr. Simi Zerine Sleeba

Department of Electronics and communication Engineering, Rajagiri School of Engineering Technology (RSET) - Autonomous, Kochi, 682039, Kerala, India.

Abstract

Multiplication is a critical operation in CNN acceleration, and it directly affects the speed and power efficiency of computation. This paper describes an optimized pipelined multiplier that combines Radix-4 Booth encoding with a Wallace tree reduction and Carry-Lookahead Adder (CLA). Comparing with traditional Booth multipliers, this approach uses pipelined Carry-Save Adders (CSAs) to minimize critical path delay. This proposed hybrid architecture is modeled using Verilog Hardware Description Language and synthesized on ZYNQ XC7Z020 SoC. Synthesis result show that pipelined booth-Wallace multiplier achieves 42.2% delay reduction when compared with a conventional booth radix-4 multiplier. The proposed design contributes performance improvement at the cost of 5.8% increase in area

Keywords: Pipelined architecture, Hardware accelerator, Booth multiplier, Wallace tree reduction technique

1 Introduction

Convolutional Neural Networks (CNNs) play major role in applications like facial recognition, medical imaging, or autonomous vehicles. This process needs a vast number of multiplication operations. For example, a single image processed through ResNet-50 requires 3.8 billion multiply-accumulate (MAC) operations [1]. Each convolutional layer in a CNN performs cross-correlations between input feature and kernels and multiplication operations account for majority of these computations. As AI models

become more complex, the demand for fast and energy-efficient multipliers is greater . Traditional processors have been unable to need this computational demand. Current Booth multiplier designs are compact. But it imposes some limitations on CNN computations due to the sequential nature of arithmetic operation .This results in considerable delays from the propagation of signals through the carry chains, thereby increasing critical path delay. They also lack the ability to take advantage of parallelism to operate on various parts of the network simultaneously. Recent studies show that multiplier units can consume up to 40% of the power for CNN accelerators . While recent designs have shown improved speed using Wallace tree structures, these have introduced new trade-offs between depth of pipelined stages and area .Through many existing designs, we can force designers into a binary choice of maximizing speed or having a feasible design chip. The proposed hybrid multiplier design is an opti- mized pipelined multiplier architecture that combining Booth encoding, Wallace tree reduction, and carry-lookahead addition.

The rest of the paper is organized as follows. Section 2 discusses previous works related to the hardware implementation of the Radix 4 algorithm and integrating into CNN. Section 3 background theory of the theory of the Radix 4 Booth multi- plier. Section 4 introduces the proposed Radix 4 - Wallace tree hybrid structure followed by the result and discussion in Section 5.Section 6 concludes this paper.

2 Related Works

The advancement in digital circuit design presents Modified Booth Multiplier (MBM) as an efficient method to perform fast multiplication. Researchers have optimized MBM to improve its performance by including Wallace Tree structure. This method can significantly reduce the number of partial products for higher order multiplication bits . The FPGA implementation with improved adder can reduce the partial product effectively [2]. Delay and power consumption decreases using partial products using Booth encoding [3]. Seunghyun Park and Daejin Park created a bit-separable radix-4 Booth multiplier that reduces computations on-the-fly with 68% lower power and 47% faster performance [4 . Muhammad Hamis Haider and Seok-Bum Ko combined Booth encoding with Power-of-Two (PO2) quantization and achieved a 30.77% reduction in CNN .The paper obtaining a 93.2% reduction in energy and ideal for edge devices. These studies confirm that Modified Booth Multiplier design is a good choice for low-power and high speed computing [5].

Beura et al. created an inexact 4:2 compressor for Baugh-Wooley multipliers. This design reduces area by 22% and power by 32.2%, while still having acceptable levels of accuracy [6]. Guem and Kim developed a variable precision Booth multiplier that can operate in either 8-bit or 16-bit mode. This design reports a 25% increase in throughput for 16-bit operations. The trade-off between precision and power or area efficiency can be particularly useful in real-time systems [7].

FPGA implementations enhance the MBM's with parallel processing and reconfigurability. Wang et al. used Booth-optimized quantization to implement a YOLOv4 as a bitstream accelerator in their FPGA system. This implementation 72.5x faster than

CPUs [8]. Additionally, Kim and Choi implemented clock gating and hardware module processing in FPGA for mobile applications .This work reports a reduction in power by 16%, and an increasing in throughput by 15%. This work highlights the importance of an FPGA in deploying Booth multipliers in high-speed and energy-aware computing. To address hardware reliability, the recent work focuses on counteracting transistor aging and timing violations [9]. Suvarna et al. used Adaptive Hold Logic (AHL) to restore performance in radix-4 Booth multipliers. The design suffered timing violations due to negative bias temperature instability (NBTI) [10]. In the architecture, Sonbul et al. designed a flexible 4096-bit Booth multiplier for polynomial operations. This design achieving speeds of 523 MHz while providing an effective energy efficiency of

1.34× over prior design. This ensures reliable performance when using this hardware in long applications [11].

A comparative study reveals the priority of the MBM over traditional multipliers. Adiono and Herdian showed that radix-8 Booth multipliers have a delay of 4.145 ns versus 5.203 ns for radix-4. Also with power dissipation reduced by 21 % [12]. Ragini and Neerajakshi showed that Dadda multipliers based on Booth principles have 27.67% lower delay and 35.34% better power delay products than array multipliers. This works suggesting that the MBM has an important place in today's VLSI systems
[13].

Cheng et al. introduce an energy-efficient sparse CNN accelerator that employs a pre-encoding radix-4 Booth multiplier to enhance energy efficiency by minimizing redundant partial products. Given the slight increase in computation, this energyefficient accelerator delivers significant power reductions achieving 7.0325 TOPS/W with 50% sparsity and 14.3720 TOPS/W with 87.5% sparsity, respectively [14].

Similarly, several papers suggested the use of approximate multipliers for CNNs with different bit-widths in order to provide a trade-off between area, delay, and accuracy. When approximating the multipliers by truncating least significant bits, power efficiency was noticeably improved, with reported accuracy being 97% or better, which showed feasibility for energy-efficient CNN accelerators [15]. Liu et al. also proposed a CNN hardware accelerator to reduce power consumption and memory accesses by reordering on-chip data, which reduced off-chip memory accesses by 82.9%. Their architecture's data path provided an improved power-reduction approach with the integration of PE arrays, multi-level memory, and a data reuse module that realized 78.75% total power savings [16]. Collectively the papers show that various approaches in optimizing CNN hardware through structural-sparsity, approximate computing, and memory efficient architectures can provide energy-efficient inference for deep learning.

2.1 Motivation

Though highly efficient, a traditional Booth multiplier utilizes significant critical path delay and power. This negatively impacts its usefulness in CNN for high-speed computing. Most current designs target either power, area reduction or speed optimization. Previous studies indicate that Booth encoding and Wallace tree reduction of the partial products result in better performance [2] . For example, a 16×16 multiplication using Radix-2 Booth encoding generates 16 partial products, whereas Radix-4 Booth encoding reduces this to only 8 partial products. However, the inherence of

pipelining within multi-bit CSAs and pipelined Wallace trees has been largely unexplored.This work attempts to address these gaps with a pipelined Booth-Wallace multiplier that utilizes CLA. The redesigned pipelined Booth-Wallace multiplier takes advantage of pipelined CSAs to reduce critical path delays. It enhances clock throughput and energy efficiency. The proposed design is implemented on FPGA.

3 Background

3.1 Radix-4 Booth Algorithm

The Booth algorithm was first introduced by Andrew Donald Booth in 1950 [17]. It is an efficient hardware method for multiplying two binary numbers in two's complement format. This technique reduces the number of partial product by half. The Radix-4 Booth encoding, the multiplier into overlapping (Q[2i 1:2i-1]) windows of bits . The radix-4 booth encoding is shown in table 1 . The groupings of three-bits, determines one of the possible operations (either: 0, $\pm$M, or $\pm$2M). (Where 2M represents a combinational multiplier being executed along with a one-bit left-shift operation to M). For example a three-bit window of 100 would denote 2M, while 011 would select 2M. This scheme is effective because it reduces the number of partial products from N for an N-bit multiplier, to N/2 a partial products. This will effectively speeds up Multiply-Accumulate (MAC) operations in CNN. Once the partial product generated, Wallace tree compression (Section 3.2) generates the total number of bits from the partial product stage and the adds them using carry-lookahead (Section 3.3). In addition, this algorithm inherently handles signed numbers without preprocessing.

Table 1 Radix-4 Booth Encoding Table

Multiplier Bits ($Q_{2i+1}Q_{2i}Q_{2i-1}$)	Operation	Partial Product	Action
000	0	0	No operation
001	$+M$	M	Add multiplicand (M)
010	$+M$	M	Add multiplicand (M)
011	$+2M$	$M \ll 1$	Add 2$\times$ multiplicand
100	$-2M$	$-(M \ll 1)$	Subtract 2$\times$ multiplicand
101	$-M$	$-M$	Subtract multiplicand (M)
110	$-M$	$-M$	Subtract multiplicand (M)
111	0	0	No operation

3.2 Wallace Tree Reduction Principles

The Wallace tree is an efficient hardware structure for partial product reduction. It achieves reduced complexity by making use of carry-save adders (CSAs). It also compresses N partial products into only two terms (sum, carry) in log3/2 N stages [18] [20]. This logarithmic reduction in time delay can be beneficial to a CNN accelerator application. Pipelined registers can also be utilized to route data strategically between

the number of compression stages . To achieve the goals of throughput and clock speed (up to 1 operation per cycle), which are necessary for real-time image processing.

3.3 Carry-Save Adders (CSA)

CSA enhance the speed of multiplication by minimizing the carry propagation delay while performing the accumulation of the partial products. Instead of allowing carries to propagate immediately. CSA keep them separate in a parallel form [19] .That reduces the total addition time in a multi-operand addition. A CSA consists of multiple full adders that operate in parallel. Each of the full adders has three input bits and produces two output bits:

$$S_i = A_i \oplus B_i \oplus C_i \quad \text{(Sum Output)} \tag{1}$$

$$C_{i+1} = (A_iB_i) + (B_iC_i) + (C_iA_i) \quad \text{(Carry Output)} \tag{2}$$

The sum is saved, and the carry is sent on to the next stage without losing speed. This helps to reduce many partial products quickly. CSAs are often used in Wallace Tree multipliers to add many numbers efficiently.

3.4 Carry-Lookahead Adder (CLA)

CLA accelerate the final addition in the multiplier by performing the calculation of carry signals in parallel. This reduces the total delay from $O(N)$ to $O(1)$ using hierarchical propagate-generate logic [19] . Each bit position produces two important signals:

$$G_i = A_i \cdot B_i \quad \text{(Carry Generation)} \tag{3}$$

$$P_i = A_i \oplus B_i \quad \text{(Carry Propagation)} \tag{4}$$

Both signals go into a multiple-level lookahead unit which computes all carries.

4 Proposed Architecture

4.1 Overall Structure

The proposed multiplier architecture combines Booth Radix-4 encoding, Wallace tree reduction, and CLA. The figure 1 shows the conceptual block diagram of the Booth-Wallace multiplier. The process begins with two inputs, as multiplicand and multiplier respectively. The multiplicand is converted to two's complement form if it is negative to perform signed multiplication. Then, the Booth Radix-4 encoders convert the multiplier in overlapping 3-bit windows and produces the partial products. The partial products are then efficiently compressed in parallel by a Wallace tree adder that consists of CSAs . Finally, a CLA is used to combine the last two terms into a result, eliminating the delayed propagation of the carry.

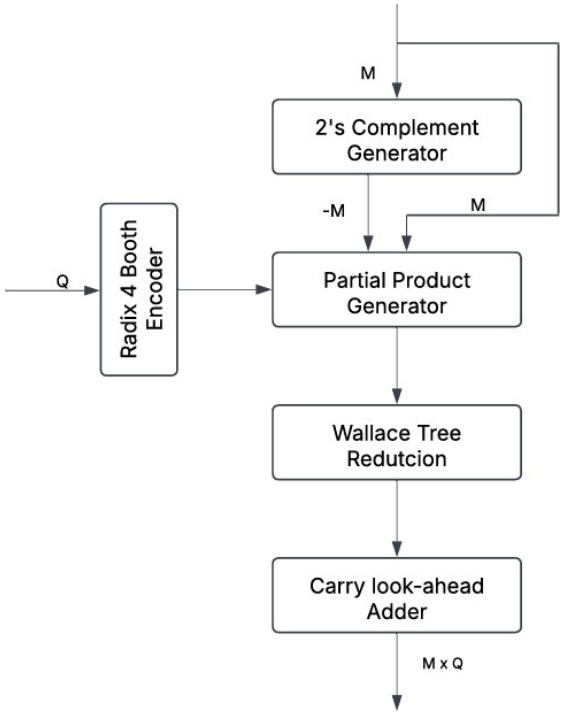

Fig. 1 Conceptual block diagram of the Booth-Wallace multiplier

4.2 Proposed Design

This figure 2 shows a stage-wise pipeline structure of a 16-bit Booth multiplier, where the Booth encoder, partial product generator, multiple CSA stages, and final carry-lookahead addition all take place sequentially. Registers are present between stages to allow for deep pipelining and high throughput.

Figure illustrates 3 how data flows entirely through the proposed multiplier that uses Radix-4 Booth encoding along with eight partial products, Wallace tree reduction using CSAs, and pipelined stages for efficient processing in hardware. The final sum is calculated using a 32-bit CLA, and the result is stored immediately after.

The first stage (A) loads the 16-bit multiplicand (M) and multiplier (Q) into input registers (M REG and Q REG). The second stage employs a Radix-4 Booth encoder that processes the multiplier and generates eight partial products (PP1–PP8) by inspecting overlapping 3-bit windows. These partial products are then passed to the Wallace tree reduction stage (stages 3–6) for progressive compression using CarrySave Adders (CSAs). The Carry-Save Adder stages CSA1 and CSA2 sum groups of three partial products (PP1–PP3 and PP4–PP6, respectively), generating partial sums (PS) and carry outputs (PC), which are left-shifted to align weights. The CSA3 stage combines one of the intermediate results from the previous CSA with the remaining

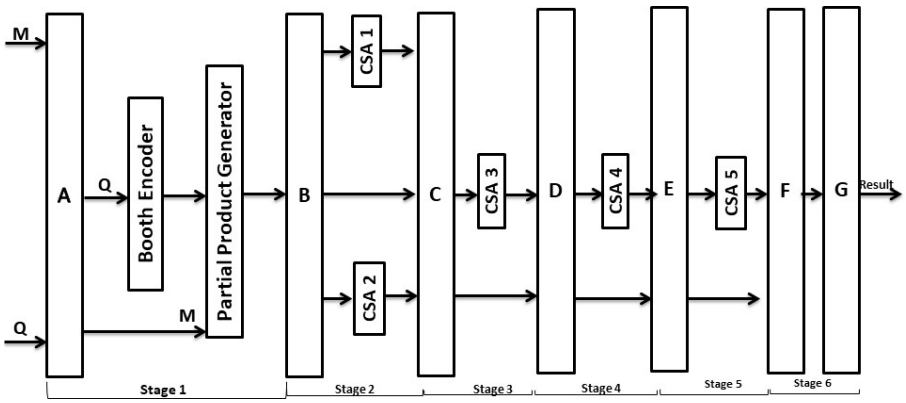

Fig. 2 Pipelined Booth Multiplier Architecture

The figure 3 shows a 16-bit $\times$ 16-bit pipelined multiplier designed using Booth encoding and Wallace tree reduction techniques. The multiplier consists of pipeline stages, maximizing throughput. The first stage loads the 16-bit multiplicand and multiplier into input registers (M REG and Q REG). The second stage employs a Radix-4 Booth encoder that processes the multiplier and generates eight partial products (PP1–PP8) by inspecting overlapping 3-bit windows. These partial products are then passed to the Wallace tree reduction stage (stages 3–6) for progressive compression using CarrySave Adders (CSAs). The Carry-Save Adder stages CSA1 and CSA2 sum groups of three partial products (PP1–PP3 and PP4–PP6, respectively), generating partial sums (PS) and carry outputs (PC), which are left-shifted to align weights. The CSA3 stage combines one of the intermediate results from the previous CSA with the remaining partial products (PP7 and PP8), while the CSA5 stage further combines the results to produce two final terms: a partial sum (S3 PS) and a carry (S3 PC). The final addition is then completed using a 32-bit Carry-Lookahead Adder (CLA), and the resulting 32-bit product is stored in the output register (RESULT REG) during the seventh pipeline stage, completing the multiplication operation. This design is efficient in both speed and area, leveraging the parallelism provided by the Wallace tree and the pipelining structure to minimize delays on critical paths.

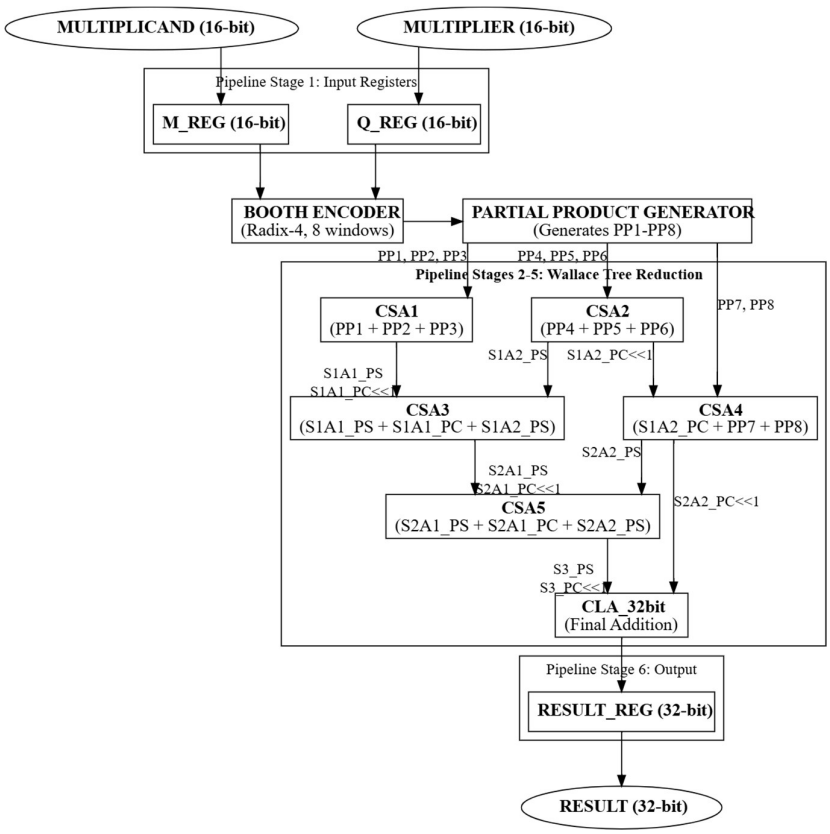

Fig. 3 Block Diagram of proposed hybrid multiplier

5 Results and Discussion

5.1 Simulation Result

The implemented pipelined Booth multiplier was synthesized and routed on a ZYNQ XC7Z020 SoC using Vivado 2023.1. The worst-case timing path was analyzed to determine the maximum achievable clock frequency. The functional correctness of the design was verified using timing simulations. Figure 4 shows the simulation output for the Wallace Tree Reduction which shows the optimal and systematic approach to reduce the partial products. Figure 5 shows the simulation outputs from the Radix 4 Booth Multiplier and the process of running the partial products and final output. The simulation confirms that the design correctly performs multiplication within the expected number of clock cycles.

Figure 6 shows the schematic of the designed Radix-4 + Wallace Tree. It shows the dataflow and critical blocks. The input operands $M[15:0]$ and $Q[15:0]$ pass through

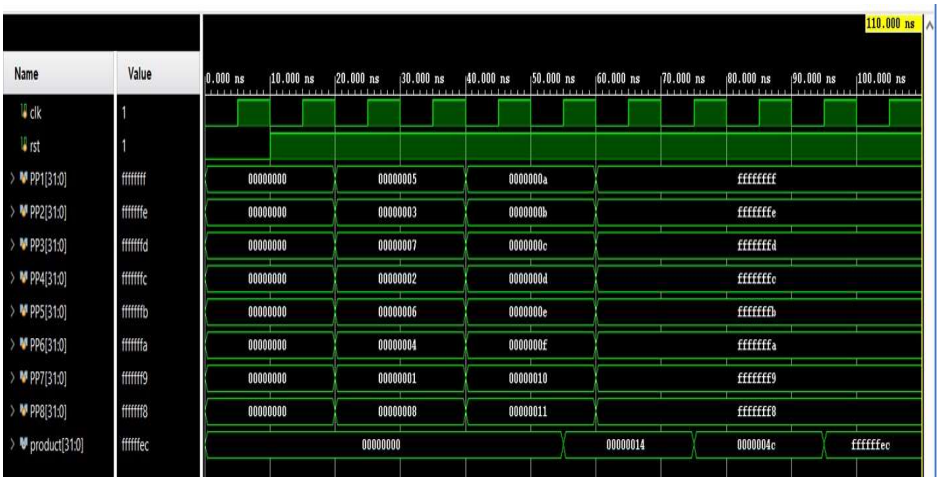

Fig. 4 Simulation result of Wallace Tree Reduction stages (Pipline Stage 2- 5)

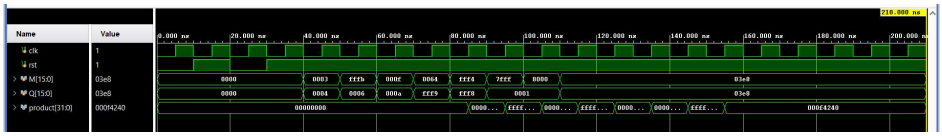

Fig. 5 Simulation result of proposed hybrid multiplier

Asynchronous registers (*Q_reg_reg*[15 : 0]) for synchronization purposes during the operation of the core.

The Booth encoder generates partial products *PP* 2[31: 0], *PP* 3[31: 0], and *PM* [31 : 0], which the core collects and sums to produce the final product *product*[31 : 0].

In addition, clear signals (CLR) and reset signals (rst) support robust initialization. The modular features in the design improve scalability for higher-speed arithmetic applications.

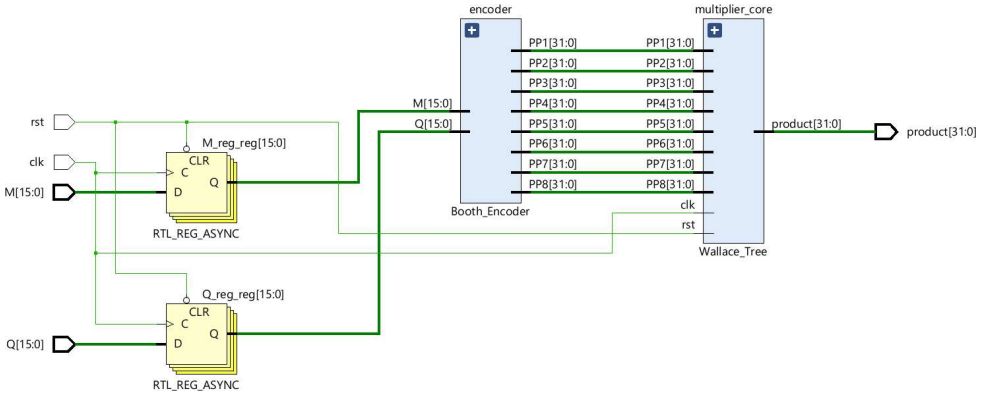

Fig. 6 Schematic Diagram of Radix 4 Booth- Wallace multiplier

5.2 Timing Analysis

The critical path delay, defined as the longest combinational delay between two sequential elements, was computed as:

$$T_{\text{critical}} = T_{\text{clk}} - \text{Slack} = 10\ \text{ns} - 2.865\ \text{ns} = 7.135\ \text{ns}$$

This suggests that the maximum achievable operating frequency of the design is:

$$f_{\max} = \frac{1}{T_{\text{critical}}} = \frac{1}{7.135\ \text{ns}} \approx 140\ \text{MHz}$$

The analysis confirms that the proposed pipelined Booth multiplier operates within the expected timing constraints, achieving an operating frequency of approximately 140 MHz on the selected FPGA. Further optimization of logic and routing resources may enhance this performance.

5.3 Performance Comparison of Multipliers

To evaluate the efficiency of the proposed pipelined Booth-Wallace multiplier, a comparative analysis was performed with a conventional Modified Booth Multiplier. The results are summarized in Table 2 . From the table, it is evident that the Modified Booth-Wallace multiplier achieves a significant speed improvement, reducing the delay from 12.34 ns to 7.135 ns, thereby increasing the maximum operating frequency to 140 MHz. This improvement comes at the cost of additional flip-flops (316 FFs) and LUTs (419 LUTs), indicating a trade-off between speed and resource utilization.

Table 2 Comparison between Radix-4 Booth Multiplier and Radix-4 Booth + Wallace Tree Multiplier

Parameters	Radix 4 Booth Multiplier	Radix 4 Booth + Wallace Tree Multiplier
Delay (ns)	12.34	7.135
No of LUT	338	419
No of FF	64	316
Power (W)	0.138	0.143

6 Conclusion

This work successfully created an improved pipelined Booth-Wallace multiplier that significantly improves Radix 4 design .With a substantial reduction of 42.2 % delay decrease and 2.5 $\times$ increased operating frequency with a 5.8% increase in total power and 24% area overhead. The new architecture's key contribution is its 7-stage pipeline structure that encompasses the Booth encoding, Wallace tree reduction and CLA addition. It can be effectively used for FPGA-based CNN accelerators and any real time signal applications that require high speed and low power. Future work will focus on using dynamic voltage scaling for thermal-aware optimization, precision-scalable operation to automate management of mixed-precision neural networks, and an ASIC implementation for energy efficiency. The scalability of the design beyond 16-bits also affords options for high-performance computing systems.

References

[1] M. Shafiq and Z. Gu, "Deep residual learning for image recognition: A survey," *Applied Sciences*, vol. 12, no. 18, p. 8972, 2022.

[2] N. Sharma and R. Sindal, "Modified Booth multiplier using Wallace structure and efficient carry select adder," *International Journal of Computer Applications*, vol. 68, no. 13, 2013.

[3] A. S. Mokhtar, N. Zahari, C. S. Ping, M. Mustapha, N. Ismail, and A. S. Ismail, "Implementation of modified Booth-Wallace tree multiplier in FPGA," *Journal of Computer Science & Computational Mathematics*, vol. 11, pp. 49–52, 2021.

[4] S. Park and D. Park, "Bit-separable radix-4 Booth multiplier for power-efficient CNN accelerator," in *Proceedings of the 2024 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, IEEE, 2024, pp. 1–6.

[5] M. H. Haider and S.-B. Ko, "Booth encoding-based energy efficient multipliers for deep learning systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 6, pp. 2241– 2245, 2023.

[6] S. K. Beura, B. P. Devi, P. K. Saha, and P. K. Meher, "Design of a novel inexact 4:2 compressor and its placement in the partial product array for area, delay, and power-efficient approximate multipliers," *Circuits, Systems, and Signal Processing*, vol. 43, no. 6, pp. 3748–3774, 2024.

[7] D.-H. Guem and S. Kim, "Variable precision multiplier for CNN accelerators based on Booth algo- rithm," *International Journal on Advanced Science, Engineering & Information Technology*, vol. 13, no. 3, 2023.

[8] Z. Wang, H. Li, X. Yue, and L. Meng, "Briefly analysis about CNN accelerator based on FPGA,"
Procedia Computer Science, vol. 202, pp. 277–282, 2022.

[9] V. H. Kim and K. K. Choi, "A reconfigurable CNN-based accelerator design for fast and energy- efficient object detection system on mobile FPGA," *IEEE Access*, vol. 11, pp. 59438–59445, 2023.

[10] S. Suvarna and K. Rajesh, "A modified architecture for radix-4 Booth multiplier with adaptive hold logic," *International Journal of Students' Research in Technology Management*, vol. 4, pp. 01–05, 2016.

[11] O. S. Sonbul, "A flexible hardware accelerator for Booth polynomial multiplier," *Applied Sciences*, vol. 14, no. 8, p. 3323, 2024.

[12] T. Adiono, H. Herdian, S. Harimurti, and T. A. M. Putra, "Design of compact modified radix-4 8-bit Booth multiplier," *International Journal on Electrical Engineering and Informatics*, vol. 12, no. 2,

pp. 228–241, 2020.

[13] K. Ragini, “Design and implementation of 4x4 bit multiplier using Dadda algorithm,” *JETIR*, vol. 6, no. 6, 2019.

[14] Q. Cheng, L. Dai, M. Huang, A. Shen, W. Mao, M. Hashimoto, and H. Yu, “A low-power sparse convo- lutional neural network accelerator with pre-encoding radix-4 Booth multiplier,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 6, pp. 2246–2250, 2022.

[15] K. Shirane, T. Yamamoto, and H. Tomiyama, “A design methodology for approximate multipliers in convolutional neural networks: A case of MNIST,” *International Journal of Reconfigurable and Embedded Systems*, vol. 10, no. 1, p. 1, 2021.

[16] Y. Liu, Y. Zhang, X. Hao, L. Chen, M. Ni, M. Chen, and R. Chen, “Design of a convolutional neural network accelerator based on on-chip data reordering,” *Electronics*, vol. 13, no. 5, p. 975, 2024.

[17] A. K. Dhumal and S. Shirgan, “Comparison between radix-2 and radix-4 based on Booth algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 12, 2016.

[18] A. M. Ghorpade and A. M. Muchandi, “Multiplier design using carry save adder,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 5, 2016.

[19] R. A. Javali, R. J. Nayak, A. M. Mhetar, and M. C. Lakkannavar, “Design of high speed carry save adder using carry lookahead adder,” in *Proceedings of the International Conference on Circuits, Communication, Control and Computing*, pp. 33–36, 2014, IEEE.

[20] S. Vaidya and D. Dandekar, “Delay-power performance comparison of multipliers in VLSI circuit design,” *International Journal of Computer Networks & Communications (IJCNC)*, vol. 2, no. 4, pp. 47–56, 2010.