

Design and Performance Evaluation of an Optimized Signed Booth Multiplier Using Verilog HDL

Shamsiah Suhaili^{1*}, Awang Muhd Azizul Haqem Awang Abdul Rahman², Norhuzaimin Julai¹, Maimun Huja Husin¹, Asrani Lit¹, Nordiana Rajae¹ and Kuryati Kipli¹

¹Electrical & Electronic Engineering Department, Faculty of Engineering, UNIMAS, 94300 Kota Samarahan, Sarawak, Malaysia

²Maltimur Resources Sdn Bhd, Northern Coastal Highway Project Main Office, 93450 Kuching, Sarawak, Malaysia

Abstract. The multiplication of signed integers is a fundamental arithmetic operation widely used in embedded systems, digital signal processing, and machine learning accelerators. This paper presents the design, synthesis, and FPGA implementation of an optimized signed Booth multiplier using Verilog Hardware Description Language (HDL). The proposed design employs the Booth algorithm to reduce the number of partial products through efficient encoding of the multiplier, followed by addition, subtraction, and arithmetic shift operations, thereby lowering hardware complexity while maintaining computational accuracy. To strengthen implementation validity, the design was functionally verified using simulation tools and further synthesized using Synopsys Design Compiler to evaluate timing performance and optimization capability. Timing analysis confirmed successful closure with a positive slack of 0.05 ns under a 5 ns clock period, indicating that all timing constraints were satisfied without violations. The Booth Multiplier was successfully designed and implemented using Verilog HDL in Quartus II, achieving a maximum operating frequency of 245.82 MHz, demonstrating efficient hardware performance on the target device. Timing analysis using Synopsys Design Compiler with a 5 ns period confirms timing closure with a positive slack of 0.05 ns under the specified operating conditions. In terms of hardware utilization, the design occupied only 21 Adaptive Logic Modules (ALMs) out of 56,480 available resources and used 27 registers, highlighting its compact resource footprint. The proposed Booth multiplier demonstrates competitive timing performance with efficient area utilization. These results confirm that the developed architecture provides a reliable and resource-efficient solution for signed integer multiplication, making it suitable for high-speed real-time digital systems and FPGA-based arithmetic accelerators.

* Corresponding author: sushamsiah@gmail.com

1 Introduction

Signed integer multiplication is a core arithmetic operation in modern digital systems and is extensively applied in embedded platforms, machine learning accelerators, image processing, and system-on-chip (SoC) architectures. Efficient multiplier design is essential for achieving high computational throughput, particularly because signed arithmetic must accurately represent both positive and negative values using two's complement representation. As computational demands and system complexity continue to grow, there is an increasing need for multiplier architectures that provide high-speed operation while maintaining efficient hardware utilization. This study focuses on the design and FPGA implementation of a Booth algorithm-based signed multiplier, which reduces partial products and simplifies hardware complexity.

In addition to performance optimization, the development of efficient arithmetic units supports sustainable engineering practices aligned with SDG 9 (Industry, Innovation, and Infrastructure) and SDG 12 (Responsible Consumption and Production). Energy-efficient hardware contributes to reduced power consumption and improved computational sustainability, which is critical in modern applications such as robotics, renewable energy systems, and intelligent automation.

Several multiplier architectures have been proposed in the literature to address performance requirements. Matrix multipliers provide high throughput and are suitable for large-scale computations, such as neural networks, but typically require substantial hardware resources, making them less suitable for resource-constrained systems. Logarithmic multipliers achieve faster multiplication through logarithmic transformations; however, they suffer from precision loss and increased complexity when applied to signed number systems. Vedic multipliers, derived from ancient Indian arithmetic techniques, offer high-speed unsigned multiplication but introduce significant overhead for signed integer operations, limiting their effectiveness.

Given these limitations, the Booth algorithm presents an attractive solution for signed integer multiplication. By encoding the multiplier to reduce redundant operations, Booth multiplication decreases the number of partial products and improves efficiency. This makes it particularly suitable for FPGA-based implementations where speed and hardware efficiency are critical. The proposed design demonstrates the effectiveness of the Booth algorithm in meeting the demands of modern digital systems.

This study contributes a refined arithmetic unit in Verilog for signed integer multiplication that can be realized on an FPGA hardware platform to enhance the overall response of embedded systems and many other applications where multiplication of signed values dominates the computation process. This research will be useful in areas that require high speed and efficiency of computations, low latency, and balanced use of resources for real-time processing, cryptography and machine learning. Several previous research have been analysed [1-7]. Based on these research, three objectives have been developed, which are to identify the literature for signed multiplication method, to develop a method to solve signed multiplication problem and to ensure that the designed method provide accurate multiplication results for signed and unsigned integers.

This study introduces a refined Verilog HDL-based multiplier architecture for FPGA realization, ensuring high efficiency in signed arithmetic operations. The contribution also includes a complete digital design flow for the Booth multiplier, starting from Verilog HDL design and verified through Functional verification in VCS to ensure correctness at RTL level. The architecture is then validated through Logic synthesis in Design Compiler, where Timing optimization with positive slack confirms that all constraints are satisfied under the target clock period. Final hardware realization is achieved via FPGA implementation in Quartus, demonstrating practical deployment on FPGA hardware. The algorithm's inherent

efficiency enables the design to achieve high operating frequency while maintaining a compact and structured hardware architecture. This makes the Booth multiplier suitable for real-time applications such as embedded systems and digital signal processing, where fast and reliable signed arithmetic operations are essential.

1.1 Booth Multiplier Algorithm

The Booth algorithm is an efficient technique for performing signed integer multiplication in binary systems, particularly suited for two’s complement representation. Unlike conventional multiplication methods that generate a partial product for every bit of the multiplier, the Booth algorithm reduces the number of partial products by encoding consecutive bits of the multiplier. This reduction leads to lower hardware complexity and improved computational efficiency, making the algorithm well suited for FPGA-based arithmetic units.

1.1.1 Principle of operation

The Booth algorithm examines two adjacent bits of the multiplier at each step: the current least significant bit Q_0 and an additional bit Q_{-1} , which is initially set to zero. Based on the combination of these two bits, the algorithm determines whether to perform an addition, subtraction, or no operation on the multiplicand. The result is accumulated in an accumulator register, followed by an arithmetic right shift to preserve the sign of the partial result. The decision rules for Booth encoding are summarized in Table 1.

Table 1. Booth Encoding Rules

Q_0	Q_{-1}	Operation
0	0	No
0	1	Add Multiplicand
1	0	Sub Multiplicand
1	1	No

1.1.2 Booth Multiplier Algorithm

The Booth multiplication process for signed integers can be summarized as follows:

1. Initialize the accumulator register to zero and set $Q_{-1}=0$.
2. Inspect the pair of bits (Q_0,Q_{-1}) .
3. Perform addition or subtraction of the multiplicand to the accumulator based on the Booth encoding rules.
4. Execute an arithmetic right shift on the combined accumulator and multiplier registers.
5. Repeat the above steps for the number of bits equal to the multiplier word length.

At the completion of the iterations, the concatenated contents of the accumulator and multiplier registers form the final signed multiplication result.

The Booth algorithm offers several advantages when implemented on FPGA platforms. By reducing the number of partial products, the algorithm decreases the number of required arithmetic operations, which directly improves speed and simplifies hardware design. Additionally, the use of arithmetic shifting ensures correct handling of signed operands in two's complement format without requiring additional sign-extension logic. These characteristics make the Booth algorithm particularly effective for high-speed and hardware-efficient signed integer multiplication in digital systems.

2 Methodology

The proposed multiplier architecture was described at Register Transfer Level (RTL) using Verilog HDL. Functional verification was first carried out through simulation using testbench vectors covering positive and negative operands to ensure correct two's complement multiplication results. After functional validation, the RTL design was synthesized using Synopsys Design Compiler to optimize logic structure and evaluate timing performance. Timing reports were analyzed to confirm successful timing closure with positive slack and no setup violations. Subsequently, the synthesized design was implemented on an Intel Cyclone V FPGA using Quartus II. Hardware evaluation was performed in terms of maximum operating frequency (FMax), logic utilization, register usage, and overall resource efficiency.

The FPGA-based implementation is evaluated in terms of maximum operating frequency, power dissipation, and hardware area to assess its suitability for real-time applications such as embedded systems and digital signal processing (DSP). In addition to hardware design and simulation, extensive verification is performed to ensure the correctness of the signed multiplication operation. Functional validation is conducted using comprehensive test vectors and comparison against expected outputs. Performance characteristics of the implemented design, including clock speed and hardware utilization, are analysed to confirm the accuracy and robustness of the developed arithmetic unit. The results demonstrate that the proposed signed integer multiplier provides a reliable and practical solution for integration into modern digital systems.

To strengthen the analysis, the obtained results were compared with previously reported multiplier architectures based on delay and hardware efficiency. This complete workflow demonstrates a practical RTL-to-FPGA methodology for developing compact and high-speed signed multiplication units suitable for embedded and real-time digital systems.

2.1 Proposed Booth Multiplier

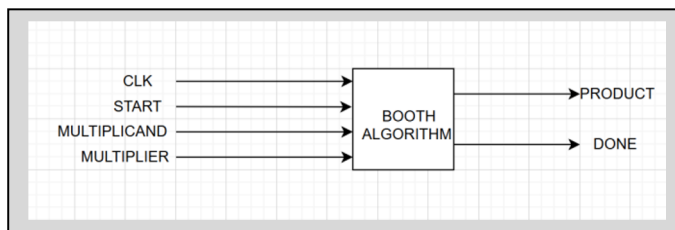


Fig. 1. Booth multiplier top level design

In the Booth multiplier architecture, the block diagram defines an organized framework that enables efficient signed multiplication through a systematic sequence of operations as shown in Figure 1. The process is synchronised by a clock (CLK) signal, while execution is triggered

by a START control signal. The multiplicand remains fixed throughout the operation and is repeatedly utilised in the computational process as required by the system. The multiplier is supplied as input and directed into the Booth encoding unit, where bit-pair evaluation determines the appropriate arithmetic action. Based on the encoded bits, the Booth processing unit performs addition, subtraction, or arithmetic shifting on the multiplicand. This strategy effectively reduces the number of partial products generated, thereby decreasing hardware complexity and improving computational speed. The multiplication proceeds iteratively, producing intermediate results that are accumulated until all multiplier bits are fully processed. Upon completion, a DONE signal is asserted, and the final computed value is presented as the PRODUCT output.

In summary, the Booth algorithm, combined with this structured block diagram, enables efficient signed integer multiplication by minimizing partial products. This results in a compact hardware design, enhanced computation speed, and reliable operation, making it particularly suitable for high-performance FPGA and embedded system applications as shown in Figure 2.

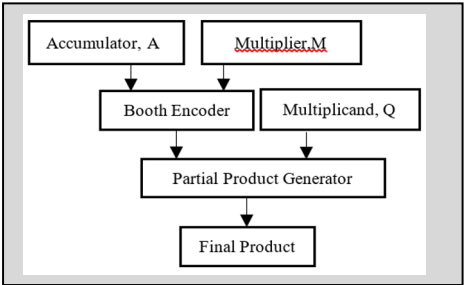


Fig. 2. Booth algorithm block diagram

Figure 3 illustrates the operation of the Booth algorithm using a flowchart. The flowchart demonstrates how Booth’s algorithm efficiently performs signed multiplication through a structured and sequential processing method. During initialization, the accumulator A is set to 0, while the multiplicand M, multiplier Q, and the counter are loaded with their respective values, with the counter set to n. The algorithm examines the least significant bit of the multiplier (Q_0) and the auxiliary bit (Q_{-1}) to determine the operation. If the bit pair is “01,” the multiplicand M is added to the accumulator A; if “10,” M is subtracted from A; and if “00” or “11,” no operation is performed. An arithmetic right shift is then applied to A, Q, and Q_{-1} , while the counter is decremented. The process repeats until the counter reaches zero, at which point the final product is stored across A and Q. This flowchart clearly illustrates how Booth’s algorithm reduces unnecessary operations while handling signed numbers in two’s complement form, enabling efficient and compact hardware design for digital systems.

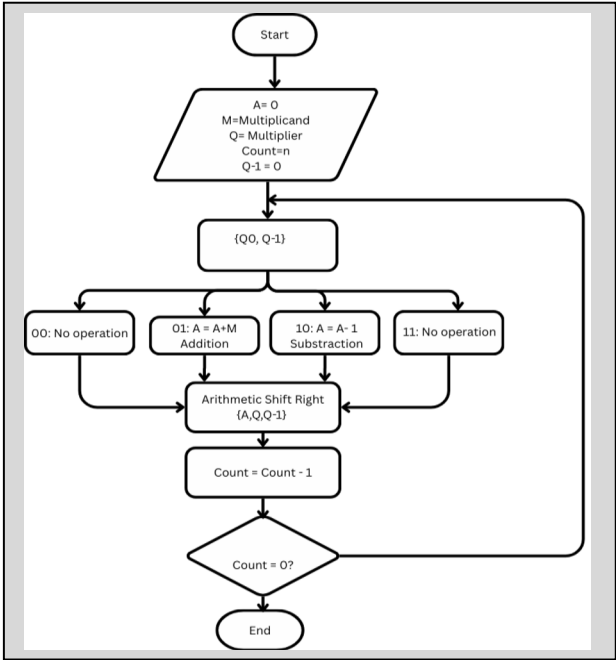


Fig. 3. Booth multiplier flowchart

3 Proposed Booth multiplier

The waveform analysis provided in the process of simulation of the Booth multiplier provides a detailed view of the algebraic flow of the process of signed integer multiplication. Since the Booth algorithm will be dealing with signed integer multiplication in two-complement form, it is of the utmost importance to test the functionality of every component of the algorithm. Here, one step at a time will be used to describe every part of the waveform, how the input operands are processed by the Booth multiplier and how it ends up with the right product. Comparison of the results is also done with real multiplication to ensure the accuracy of the simulation and that the algorithm used is correct and gives accurate results.

The waveform gives a detailed picture on the operation of Booth multiplier in two signed integers multiplication. The multiplicand is displayed at the beginning of the wave form. The multiplicand value is changing in synchronous clock (clk). Being one of the operands used in the multiplication process, the multiplicand does not disappear in the computation but rather takes part in producing partial products. And this operand is kept getting updated based on the clock cycle, which is reflected as the waveform.

The bit corresponding to a multiplier (which is a part of the Booth algorithm) is emphasized in the middle part of the waveform. This coding decreases the number of partial products required during the multiplication. The waveform graphically illustrates the treatment of bits of the multiplier bit by bit over every clock cycle showing intermediate results produced by the algorithm. The Booth algorithm switches between addition and subtractions depending on how the bit-pairs of the multiplier are coded, as can be seen by glancing at the switches across partial products.

The arithmetic operations and the shifts in Booth algorithm are made on the bits of the multiplier and each variation on the multiplier can be observed on the waveform as it passes through the clock cycle. This is done in a step wise manner, and the intermediate values are finally added to give the final product. When the multiplication process has been done, the result is displayed as waveform and this value is the product of the two signed integers

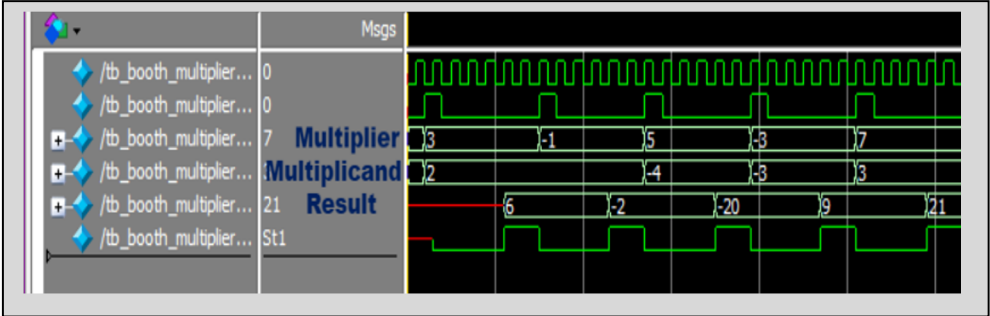


Fig. 4. Booth Multiplier Simulation Waveform using ModelSim

Figure 4 indicates the first step of the operation of the Booth multiplier that involves the processing of the multiplier 3 and the multiplicand 2 and the production of the first partial product. Multiplier is indicated in the top part of the waveform and in the middle part is multiplicand indicated where it is the value 2. These two values are the initial operands of the multiplication procedure, and they are essential in the following processes of the algorithm developed by Booth.

The algorithm used by Booth encodes the multiplier then uses the encoding to compute partial products. In this process, the initial one is checking the multiplier 3 (as expressed by 0011 in 4-bit binary type), along with considering the multiplicand 2 (as expressed by 0010 in binary type). In this step the algorithm produces a partial product which is called by multiplying the multiplier 3 by 2 and the answer is 6. This partial product of 6 is produced at the first loop of the algorithm of Booth and this proves that the multiplier has been successfully encoded and the multiplier has been properly handled.

The outcome of the $3 \times 2 = 6$ is the initial intermediate result during the multiplication process. At this step Booths algorithm has already produced first partial product and has laid the foundation of producing more partial product in the next cycles. This is one step that shows that the algorithm of Booth is doing as expected in multiplying the multiplicand and multiplier and getting the expected product. This waveform demonstrates that the multiplier 2 has been coded properly and the first multiplication computation has been performed correctly which results to the expected 6.

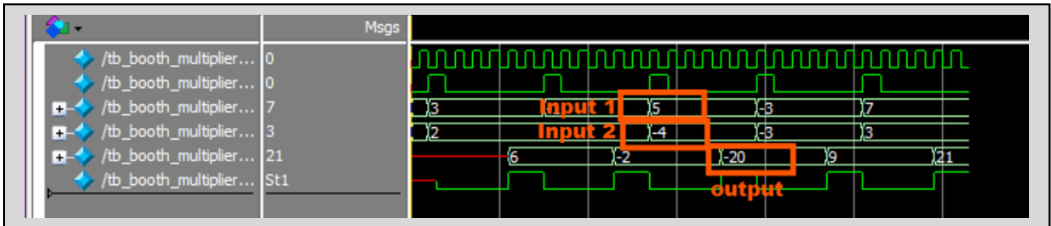
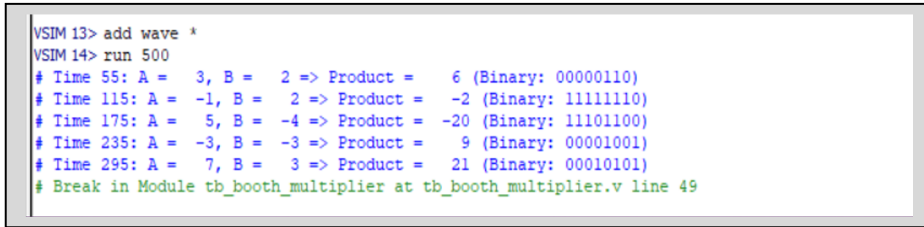


Fig. 5. Example of Booth Multiplier with result of -20

Figure 5 shows the simulation waveform of one operating cycle of the Booth multiplier for signed multiplication. In this cycle, the multiplier (Input 1) is set to 5 (0101), while the

multiplicand (Input 2) is -4 , represented in two's complement form as 1100. Based on the Booth recoding of the multiplier bits, the control logic selects the appropriate arithmetic operation on the multiplicand. The accumulation and arithmetic shifting performed during this cycle generate an output of -20 , corresponding to the correct product of $5 \times (-4)$. The appearance of this result in the waveform confirms the correct handling of signed operands and validates the functional correctness of the Booth encoding and arithmetic logic implemented in the proposed design.



```

V$IM 13> add wave *
V$IM 14> run 500
# Time 55: A = 3, B = 2 => Product = 6 (Binary: 0000110)
# Time 115: A = -1, B = 2 => Product = -2 (Binary: 1111110)
# Time 175: A = 5, B = -4 => Product = -20 (Binary: 11101100)
# Time 235: A = -3, B = -3 => Product = 9 (Binary: 00001001)
# Time 295: A = 7, B = 3 => Product = 21 (Binary: 00010101)
# Break in Module tb_booth_multiplier at tb_booth_multiplier.v line 49

```

Fig. 6. Listing Result of Booth Multiplier Testbench Simulation in VCS

In addition to ModelSim simulations, the Booth multiplier design was also simulated using the VCS tool in Synopsys as shown in Figure 6 to provide an independent verification of the functionality. VCS is a high-performance simulation environment that enables cycle-accurate verification and cross-validation of RTL designs. The VCS simulation confirmed that the multiplier correctly handles signed integer operations, producing results consistent with those obtained from ModelSim. By applying the same testbench vectors in VCS, the multiplication of both positive and negative operands generated expected decimal and binary outputs, matching the two's complement representation. For example, $A=-3$ and $B=-3$ produced 9 (binary 00001001), and $A=5$ with $B=-4$ resulted in -20 (binary 11101100), consistent with ModelSim results. This cross-verification enhances confidence in the correctness of the RTL implementation, ensuring that the Booth multiplier reliably performs signed multiplication under different simulation environments.

3.1 FMax Analysis

One of the most important parameters used in FPGA design is FMax (Maximum Frequency) which is the maximum clock frequency to which the design can be operated, and the design meets the timing requirement. The value itself is central in determining the performance of the design in general since it directly affects the rate at which FPGA can do the calculations. In the case of Booth multiplier design, we consider Fmax to be 245.82 MHz which indicates that the design will be able to operate reliably at the clock frequency of 241.6 MHz without any timing violation.

Figure 7 shows the frequency of 245.82 MHz (FMax) will calculate the maximum speed at which the Booth multiplier can work, and therefore it is able to perform the signed integer multiplication operation typical of the FPGA. Higher FMax value is usually related to faster design, and it is possible to execute arithmetic operations faster. In this, the 245.82 MHz implies that the Booth multiplier can accomplish the assigned multiplication activities at a relatively. faster rate, which is adapted to the many real-time applications, which demand the high-speed processing activities. Given that FMax is the maximum clock frequency in which the design may be run, FMax also indirectly describes how well the design is overall and how optimized it is. A design that has a greater FMax will mostly have a larger number of operations that it can perform in a single second and has a larger throughput and can perform its calculations faster. FPGA based designs require high FMax to ensure achievement of the

performance demands of the complicated tasks such as multiplication in the real time systems.

Compilation Report - booth_multiplier				
Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	245.82 MHz	245.82 MHz	clk	

Fig. 7. FMax of Booth Multiplier Design

The maximum clock frequency $F_{Max} = 245.82\text{ MHz}$ is the maximum frequency the Booth multiplier design can operate when all the timing constraints are met, and the design operates without any timing or error. This figure ensures that the design can undergo operations with this clock frequency and hence offers effective solution in terms of multiplication in FPGA applications needing fast computations. The restricted F_{Max} value that is also at 245.82 MHz means that the design will sustain this frequency even at specified voltage of 1100mV and temperature of 85°C . The observation that the limited F_{max} is the same as the F_{Max} implies that Booth multiplier structure is tolerant of environmental variations thus making the structure resistant to changes in the environment even under normal working conditions.

3.2 RTL Design Analysis

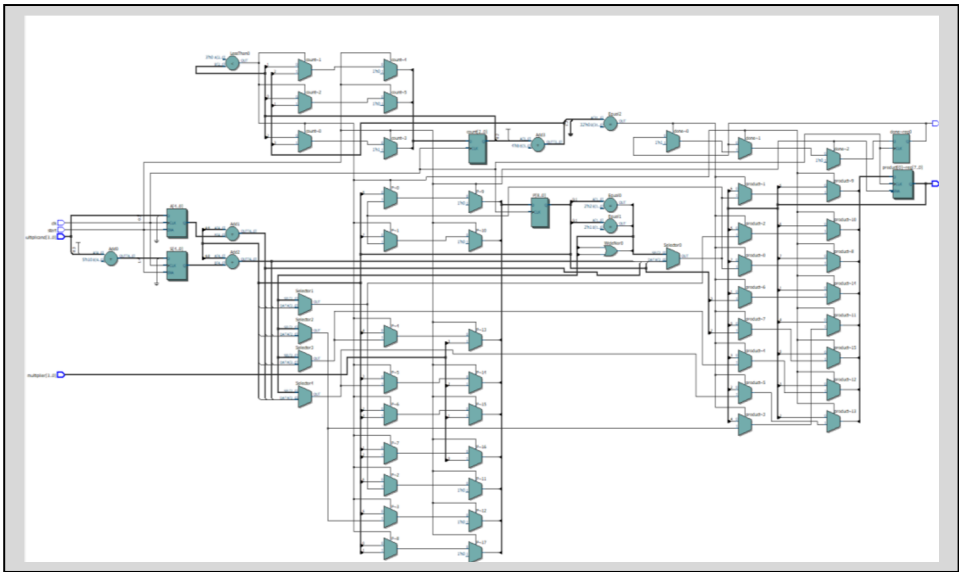


Fig. 8. FMax of Booth Multiplier Design

Figure 8 represents the RTL (Register Transfer Level) implementation of the Booth multiplier and demonstrates the digital hardware architecture which realizes the signed integer multiplication using the Booth algorithm. It was designed with simple digital elements named register, multiplexers, AND gates and adders which were integrated together to carry out the required actions which is the Booth multiplication process The Register Transfer Level (RTL) architecture of the Booth multiplier is a digital Realization (hardware implementation) of the algorithm given by Booth to multiply signed integers. The design

involves all important components which combine to effectively perform the process of multiplication. Registers are at the centre of the design, and they contain the multiplicand and the multiplier. The bits of the multiplier are processed one at a time and operations are done accordingly based on the value of the current and the previous bit. Multiplexer (MUX) is crucial during this step, since it determines what values to go through next, based on the multiplier bits state.

It also uses shift registers which store upcoming intermediate results and shifts the bits to the right when required. It is a fundamental part of Booth multiplier that a shifting operation takes place since each iteration, the algorithm divides the multiplier by two, gradually reaching the required outcome. As one pair of bits of the multiplier is reached, they are also added or subtracted (by using an adder and a subtractor) to the previous partial product of multiplicand. This determination is supported by the current and past segments of the multiplier. When the bits are equal no operation is executed but when it is different one out of addition and subtraction is done.

The running sum of the intermediate results is stored in the accumulator, and this is the partial product. Whenever a shift or an operation is completed, the result will be stored in the accumulator and multiplier shifted. This will repeat itself until all the bits of the multiplier are completed. Another key component in such an operation is the logic of control that coordinates the timing and the order of the whole process. It decides, at what time the registers are supposed to move, when the addition or subtraction is expected, and when the result is supposed to be placed in the accumulator. The implementation design realized that the Booth multiplier will support both negative and positive number calculation with the two-complement arithmetic process, thus can be used in many applications that need to multiply signed integer. The design can come up with the final product, by processing the multiplier bit wise, shifting and executing necessary arithmetic simple tasks.

3.3 Synopsys-Based Synthesis and Optimization Analysis

This section presents the synthesis and optimization outcomes of the proposed Booth multiplier design. The evaluation includes functional verification, architectural representation, and post-synthesis performance analysis in terms of area, timing, and slack. Figures 9 to 16 summarize the functional behavior, structural implementation, and timing performance of the design after synthesis. These results are essential for validating the suitability of the multiplier architecture for ASIC realization and high-speed digital applications. The Booth multiplier was synthesized and optimized using Synopsys Design Compiler, and results were validated through simulation, schematic inspection, area reporting, and timing analysis to ensure both functional correctness and performance efficiency.

Figure 9 shows the simulation waveform used to verify the functional correctness of the multiplier. The applied test input of $5 \times (-4)$ generates an output of -20 , confirming correct signed multiplication using two's complement arithmetic. The waveform also demonstrates that the Booth encoding and accumulation logic operate correctly across sign boundaries and that the implementation matches the algorithmic specification.

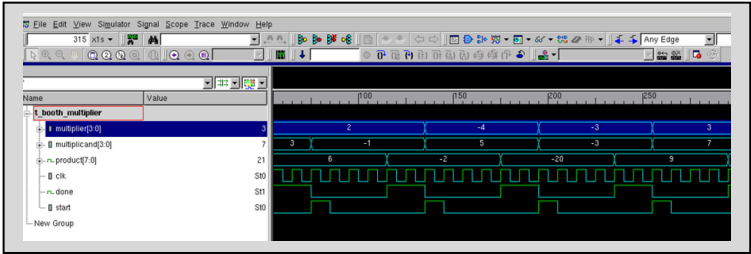


Fig. 9. Simulation Waveform of Booth Multiplier Design

The structural and hierarchical implementations are illustrated in Figure 10 and Figure 11, which depict the top-level integration and schematic design of the multiplier system. Figure 10 presents the top-level hierarchical view of the complete Booth multiplier design. It integrates all submodules into a single architecture and shows the external interface ports such as clock, reset, multiplicand, multiplier, and product output. This view emphasizes the module’s reusability and how it interfaces with surrounding system logic.



Fig. 10. Top Level Design of Booth Multiplier

Figure 11 illustrates the synthesized schematic of the Booth multiplier. The schematic presents the post-synthesis structural view of the multiplier, showing the mapped arithmetic units, shifters, adders, and control logic. This representation clarifies how the Booth algorithm is realized in gate-level components and highlights the dataflow and control signals used during multiplication. This confirms successful RTL-to-gate-level synthesis.

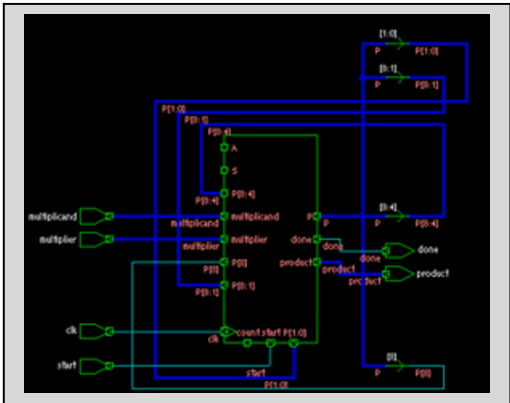


Fig. 11. Schematic Design of Booth Multiplier

Figure 12 represents the pre-optimization schematic illustrates the initial implementation prior to synthesis-driven optimizations. The design contains a larger logic network and less efficient mapping, which may increase area usage and delay. This figure serves as a reference for evaluating post-optimization improvements. It typically shows larger combinational blocks and longer unbalanced paths, which contribute to increased area and a longer critical path that limits achievable clock frequency.

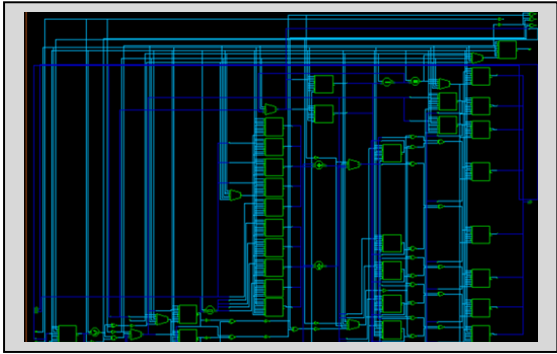


Fig. 12. Schematic Design of Booth Multiplier Pre-optimized

After optimization, the area report in Figure 13 shows a total cell area of 1620.789992 and a total design area of 1685.097103. This corresponds to an area utilization of approximately 96.2%, indicating efficient use of silicon resources with minimal overhead after optimization. These figures demonstrate that synthesis optimizations reduced the silicon footprint while preserving functionality. Timing performance is constrained at a 5 ns clock period as shown in Figure 14.

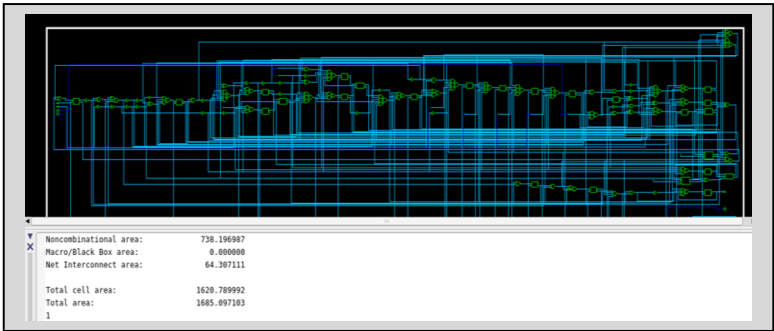


Fig. 13. Area Report of Optimized Booth Multiplier

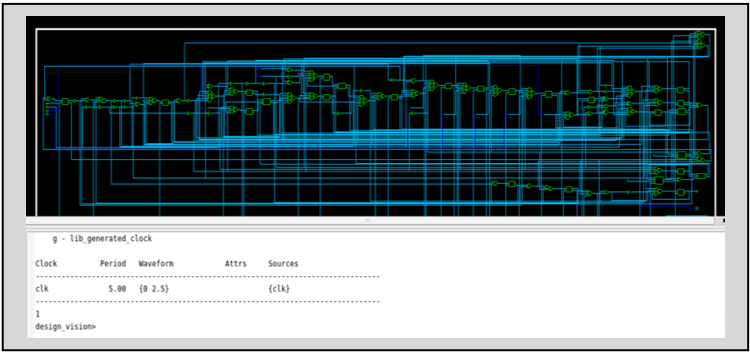


Fig. 14. Clock Report of Optimized Booth Multiplier

The optimized design achieves a positive slack of 0.02 ns as shown in Figure 15, confirming that all critical paths meet timing requirements. Since the slack is positive, all critical timing paths meet the required setup timing constraints, confirming stable operation under the defined clock condition. This indicates that the worst-case timing path completes slightly earlier than the clock period requirement. Positive slack confirms that timing constraints are satisfied with a small safety margin.

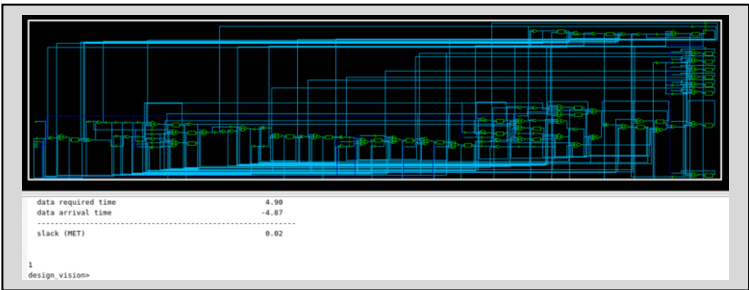
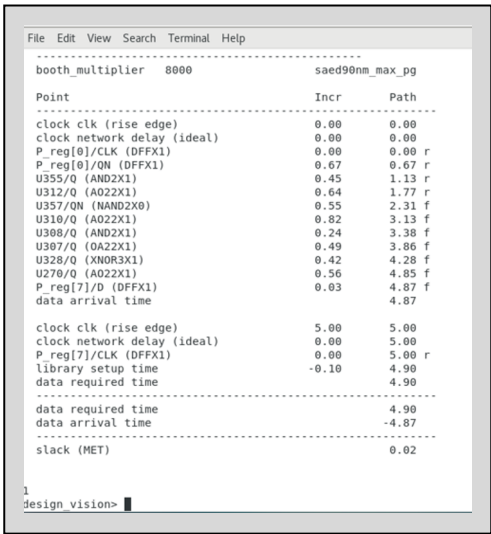


Fig. 15. Slack (MET) of Optimized Booth Multiplier

The timing report in Figure 16 confirms positive slack across timing paths and indicates successful timing closure at the specified 5 ns period. This validates that the optimized netlist meets the timing requirements imposed for the intended operating frequency. Ffurther verifies successful timing closure with no violations under the specified operating conditions. This corresponds to a target operating frequency of 200 MHz, indicating that the design is optimized for high-speed synchronous operation. Figure 16 summarizes the final timing analysis. The positive slack and absence of timing violations confirm successful timing closure. Therefore, the optimized Booth multiplier is suitable for ASIC implementation and reliable real-time processing applications.

A screenshot of a Synopsys Design Visioner window showing a timing report for a design named 'saed90nm_max_pg'. The report lists various timing points, their incremental delays, and total paths. The design is a Booth multiplier with a multiplier of 8000. The report shows a clock frequency of 5.00 MHz and a total slack of 0.02 ns. The design is implemented in a 90nm technology.

booth_multiplier 8000 saed90nm_max_pg		
Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
P_reg[0]/CLK (DFFX1)	0.00	0.00 r
P_reg[0]/QN (DFFX1)	0.67	0.67 r
U355/Q (AND2X1)	0.45	1.13 r
U312/Q (AO22X1)	0.64	1.77 r
U357/QN (NAND2X0)	0.55	2.31 f
U310/Q (AO22X1)	0.82	3.13 f
U308/Q (AND2X1)	0.24	3.38 f
U307/Q (AO22X1)	0.49	3.86 f
U328/Q (XNOR3X1)	0.42	4.28 f
U270/Q (AO22X1)	0.56	4.85 f
P_reg[7]/D (DFFX1)	0.03	4.87 f
data arrival time		4.87
clock clk (rise edge)	5.00	5.00
clock network delay (ideal)	0.00	5.00
P_reg[7]/CLK (DFFX1)	0.00	5.00 r
library setup time	-0.10	4.90
data required time		4.90
data required time		4.90
data arrival time		4.87
slack (MET)		0.02

Fig. 16. Timing Report of Optimized Booth Multiplier

In conclusion, the Synopsys-based synthesis and optimization process demonstrates that the Booth multiplier achieves correct functionality, efficient hardware utilization, and reliable timing performance. Across Figure 9 to Figure 16, the results confirm successful functional validation, approximately 96.2% area utilization efficiency, and positive timing slack, ensuring that the design satisfies the required 5 ns constraint. Overall, the optimized Booth multiplier is well-suited for high-performance FPGA and ASIC implementation.

Conclusions

In conclusion, the results obtained on the synthesis were detailed about the performance measurement such as the FMax of 245.82 MHz and the results show that the design functions well at this clock frequency. It was also identified in the analysis that the Booth multiplier is minimal in the FPGA resources. The RTL design description showed the hardware layer of the Booth multiplier giving the explanation of how the registers, multiplexers, adders and control logic interacted. It showed step-by-step application of these components to implement the Booth algorithm such that correct and efficient signed integer multiplication is possible.

This is optimized in terms of the design, hence consuming minimum resources to attain a high operational speed; a property which makes it useful in application of high-speed real time applications that demand high speed arithmetic operations. The design of Booth multiplier is efficient and reliable, which can process signed integer multiplication with high throughput and low resource requirements. The flexibility of this design and its viability in different environmental conditions makes it an appropriate design in real time FPGA based applications in industry and research. Performance evaluation is further characterized using FMax, LUT, registers, indicating efficient speed and resource utilization. Overall, the results confirm a balanced design with strong timing closure, optimized area usage, and high-frequency operation suitable for advanced digital systems. The final implementation utilized 21 Adaptive Logic Modules (ALMs) and 27 registers while achieving an operating frequency of 245.82 MHz. These improvements demonstrate a successful trade-off between area and performance, yielding a design that is both more compact and capable of operating at the target high frequency.

References

1. A. Puşcaşu, C. B. Ciobanu, O. Buiu, Systolic Array Matrix Multiplication Accelerator, in 2024 International Semiconductor Conference (CAS), Sinaia, Romania, (2024), 207-210.
<https://doi.org/10.1109/CAS62834.2024.10736842>
2. J. S, M. M, K. Kumar M, H. Krithik Roshan S, VLSI Design of Majority Logic based Wallace Tree Multiplier, in 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, (2023), 967-971.
<https://doi.org/10.1109/ICCMC56507.2023.10083704>
3. K. Naresh, Y. P. Sai, S. Majumdar, Design of 8-bit Dadda Multiplier using Gate Level Approximate 4:2 Compressor, in 2022 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Bangalore, India, (2022), 269-274.
<https://doi.org/10.1109/VLSID2022.2022.00059>
4. Priyanka C, Chiranth C Shet, Darshan M, Mahaveer Rathod, A Reconfigurable Precision Hybrid Booth-Encoded Wallace-Dadda Multiplier for Energy-Efficient DSP and AI Hardware. International Journal of Engineering Research & Technology (IJERT).**15**, 1-4 (2026).
<https://doi.org/10.5281/zenodo.19945626>
5. K Naga Lakshmi, V Sravan Kumar, R Revathi, T. Sirisha, N Pavankumar, T Uha Lakshmi, Efficient Low-Power VLSI Design of a High-Speed Baugh-Wooley Multiplier Using Verilog HDL. International Journal for Modern Trends in Science and Technology, **11**, 254-260 (2025).
<https://doi.org/10.5281/zenodo.15093953>
6. H. M. Brunda, H. Jayalaxmi, H. N. Arpitha, Anupama R. Patil, G. S. Manjushree, Comparative Analysis of Various Multipliers Based on Performance, Scientia. Technology, Science and Society, **2**, 100-108 (2025),
[https://doi.org/10.59324/stss.2025.2\(5\).08](https://doi.org/10.59324/stss.2025.2(5).08)
7. A. S. N. Mokhtar, N. Zahari, C. S. Ping, N. Ismail, A. S. Ismail, Implementation of Modified Booth-Wallace Tree Multiplier in FPGA. ZULFAQAR Journal of Defence Science, Engineering & Technology, **5**, 1-12, (2022).
<https://doi.org/10.58247/jdset-2022-0502-10>.
8. L. Renuka Devi, N. Arumugam, A. Saravanaselvan, A Review on Multiplier Architecture and Optimization. Ilkogretim Online - Elementary Education Online, **20**, 2194-2204 (2021).
<https://doi.org/10.17051/ilkonline.2021.01.242>.