Project 34: Booth Multiplier

A Comprehensive Study of Advanced Digital Circuits

By: Gati Goyal ,Abhishek Sharma, Nikunj Agrawal, Ayush Jain

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Introduction

The Booth Multiplier is a well-established algorithm that enhances the efficiency of binary multiplication by minimizing the number of arithmetic operations involved, particularly in handling signed numbers. Traditionally, binary multiplication involves generating and summing partial products, but this can be time-consuming, especially with signed binary numbers in two's complement representation. The Booth algorithm optimizes this process by encoding sequences of consecutive 1's in the multiplier, thus reducing the number of required additions and subtractions.

Unlike conventional multiplication methods, the Booth Multiplier scans pairs of bits in the multiplier and performs operations based on the changes between adjacent bits. If the pair changes from 0 to 1, the algorithm subtracts the multiplicand, and if it changes from 1 to 0, it adds the multiplicand. Sequences of repeated 1's are handled efficiently by treating them as a single operation, minimizing redundant additions or subtractions and speeding up the multiplication process. This makes Booth's algorithm particularly useful in handling numbers with long runs of 1's, which are common in two's complement numbers.

This documentation provides a detailed examination of the Booth Multiplier, outlining its underlying principles, structure, and implementation using hardware description languages like System Verilog. It highlights the advantages of Booth's algorithm, including its efficiency in reducing computation time and improving the performance of signed binary multiplication, particularly in high-performance digital systems.

In binary multiplication, both signed and unsigned binary numbers are processed to produce a binary product. For signed multiplication, Booth's algorithm is an efficient approach that eliminates unnecessary operations and accelerates the overall computation. This project aims to design and implement a Booth Multiplier in System Verilog, focusing on the algorithm's ability to handle signed numbers and improve computational efficiency.

The architecture of the Booth Multiplier utilizes sequential logic to perform multiplication by iterating through the bits of the multiplier and selectively adding or subtracting the multiplicand based on the Booth encoding. This design ensures optimal speed and resource utilization, especially in scenarios where signed numbers are involved. By executing the multiplication in a series of controlled steps, the Booth Multiplier enhances the speed of computation while maintaining accuracy.

This documentation provides a thorough overview of the Booth Multiplier's structure, operational principles, and performance evaluation through simulation. By analyzing the implementation details and conducting rigorous testing, the project demonstrates the effectiveness of the Booth Multiplier in digital circuit design, particularly in applications that require fast and efficient signed binary multiplication.

—

2 Background

The Booth Multiplier was developed to address the delays associated with carry propagation in traditional binary multipliers, such as the Shift-and-Add method. In conventional multiplication, partial products are summed sequentially, leading to performance bottlenecks, especially as the bit-width increases.

It utilizes Booth Multiplier to accumulate partial products in parallel. This approach separates carry bits from sums until the final addition stage, effectively minimizing carry propagation delays. By enabling parallel processing, CSAM significantly enhances multiplication speed, making it well-suited for high-performance applications, including digital signal processing, graphics, cryptography, and computing.

Furthermore, when integrated with Booth's algorithm, CSAM improves efficiency in handling large-bit multiplications. Booth's algorithm optimizes the multiplication process by reducing the number of partial products, and when combined with the CSAM's array structure, it enhances throughput and overall performance. The combination of these techniques allows for efficient organization and addition of partial products, leading to faster and more efficient multiplication in various computational contexts.

3 Structure and Operation

The Booth Multiplier is designed to optimize binary multiplication, particularly for signed numbers, by minimizing the number of addition and subtraction operations. Its structure and operation focus on

encoding the multiplier and handling partial products efficiently.

3.1 Key Components

- **Input Ports**: The Booth Multiplier receives two inputs, A (the multiplicand) and B (the multiplier). Each input is typically n bits wide, and both can be positive or negative in two's complement form.
- **Booth Encoder**: The Booth encoding logic examines pairs of bits (including an extra reference bit) from the multiplier B and generates control signals. Based on these signals, it decides whether to add, subtract, or skip operations on the multiplicand A.
- **Adder/Subtractor Unit**: This unit is responsible for either adding or subtracting the multiplicand A to/from the partial sum, as directed by the Booth Encoder.
- **Shift Register**: The partial product is stored in a shift register, which shifts the intermediate result by one position after each step of the operation. This allows for accumulation of partial products and ensures correct bit alignment.
- **Control Unit**: The control unit synchronizes the operations of the Booth Encoder, adder/subtractor, and shift register, ensuring the multiplication process follows the Booth algorithm.
- **Output Port**: The final output, P, is the product of the multiplication and is typically $2n$ bits wide.

3.2 Operational Steps

The operation of the Booth Multiplier proceeds as follows:

1. **Input Initialization**: The multiplicand A and multiplier B are loaded into their respective registers.
2. **Booth Encoding**: The Booth Encoder examines each pair of bits from the multiplier B and generates control signals (add, subtract, or skip) based on the bit patterns.
3. **Partial Product Accumulation**: Depending on the Booth Encoder's output, the adder/subtractor unit either adds or subtracts the multiplicand A to/from the current partial product.
4. **Shift Operation**: After each addition or subtraction, the partial product is shifted right by one position to align the bits correctly.
5. **Repeat Process**: The encoding, addition/subtraction, and shifting steps are repeated until all bits of the multiplier have been processed.
6. **Final Output Generation**: Once all bits have been processed, the final product P is output as a signed binary number, typically $2n$ bits wide.

By using Booth's encoding, the Booth Multiplier reduces the number of required operations, especially when dealing with long sequences of 1's in the multiplier. This makes it highly efficient for signed binary multiplication, ensuring faster performance in digital systems that require high-speed arithmetic.

4 Implementation in System Verilog

The following RTL code implements the Booth Multiplier in System Verilog:

Listing 1: Booth Multiplier

```
1
2   module booth_multiplier (
3     input logic clk,
4     input logic rst_n,
5     input logic start,
```

```
    input logic signed [15:0] multiplicand,
    input logic signed [15:0] multiplier,
    output logic signed [31:0] product,
    output logic done
);
    logic signed [31:0] A, Q, M;
    logic Qn;
    logic [4:0] count; // Up to 16 bits

    always_ff @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            product <= 0;
            A <= 0; Q <= 0; Qn <= 0; M <= 0; count <= 0; done <= 0;
        end else if (start) begin
            A <= 0; Q <= multiplier; Qn <= 0; M <= multiplicand; count
               <= 16; done <= 0;
        end else if (count > 0) begin
            case ({Q[0], Qn})
                2'b01: A <= A + M; // Add
                2'b10: A <= A - M; // Subtract
            endcase
            {A, Q} <= {A, Q} >> 1; // Arithmetic right shift
            Qn <= Q[0]; // Update Qn
            count <= count - 1;
        end else if (count == 0) begin
            product <= {A, Q}; // Final product
            done <= 1;         // Multiplication complete
        end
    end
endmodule
```


5 Test Bench

The following test bench verifies the functionality of the Booth Multiplier :

Listing 2: Booth Multiplier Testbench

```

  module tb_booth_multiplier;
    logic clk, rst_n, start;
    logic signed [15:0] multiplicand, multiplier;
    logic signed [31:0] product;
    logic done;

    booth_multiplier uut (
        .clk(clk),
        .rst_n(rst_n),
        .start(start),
        .multiplicand(multiplicand),
        .multiplier(multiplier),
        .product(product),
        .done(done)
    );

    initial begin
        clk = 0; rst_n = 0; start = 0;
        #5 rst_n = 1;

```

```
        // Test cases
        multiplicand = 16'sd3; multiplier = 16'sd4; start = 1; #10
            start = 0;
        wait(done);
        $display("Product of %d and %d is: %d", multiplicand,
            multiplier, product);

        multiplicand = 16'sd5; multiplier = 16'sd6; start = 1; #10
            start = 0;
        wait(done);
        $display("Product of %d and %d is: %d", multiplicand,
            multiplier, product);

        multiplicand = 16'sd7; multiplier = 16'sd3; start = 1; #10
            start = 0;
        wait(done);
        $display("Product of %d and %d is: %d", multiplicand,
            multiplier, product);

        multiplicand = 16'sd8; multiplier = 16'sd2; start = 1; #10
            start = 0;
        wait(done);
        $display("Product of %d and %d is: %d", multiplicand,
            multiplier, product);

        // Finish simulation
        $finish;
    end

    // Clock generation
    always #5 clk = ~clk;

endmodule
```


6 Advantages and Disadvantages

6.1 Advantages

- **Efficient for Signed Numbers**: The Booth Multiplier is particularly advantageous for signed binary multiplication as it handles both positive and negative numbers using two's complement representation.
- **Fewer Operations**: By encoding sequences of consecutive 1's in the multiplier, Booth's algorithm reduces the number of addition and subtraction operations, leading to faster computation, especially for numbers with long sequences of 1's.
- **Reduced Hardware Complexity**: The Booth Multiplier reduces the number of partial products, simplifying the multiplication process and requiring fewer adders than traditional array multipliers.
- **Flexibility in Handling Large Bit-widths**: The Booth Multiplier can handle large bit-width multiplications without significant increases in complexity, making it scalable and efficient for digital systems.

6.2 Disadvantages

- **Variable Number of Operations**: The number of operations in the Booth Multiplier depends on the bit patterns of the multiplier, which can lead to performance variation. In some cases, it may perform more slowly than simpler multipliers if the bit patterns require frequent operations.

- **Complex Control Logic**: Booth's algorithm requires additional control logic to handle the encoding and selection of add/subtract operations, increasing the design complexity.
- **Shifting Overhead**: The shifting of partial products during each step introduces overhead, and combined with the additional control required, can affect the overall speed for certain input patterns.
- **Higher Power Consumption**: Due to the complex control logic and the need for multiple operations and shifts, the Booth Multiplier may consume more power than simpler multipliers, which can be a disadvantage in power-sensitive applications.

7 Simulation Results

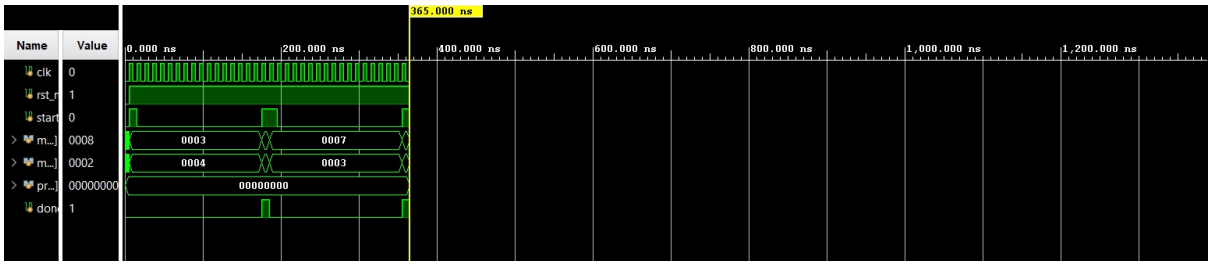

Figure 1: Simulation results of Booth Multiplier

8 Schematic

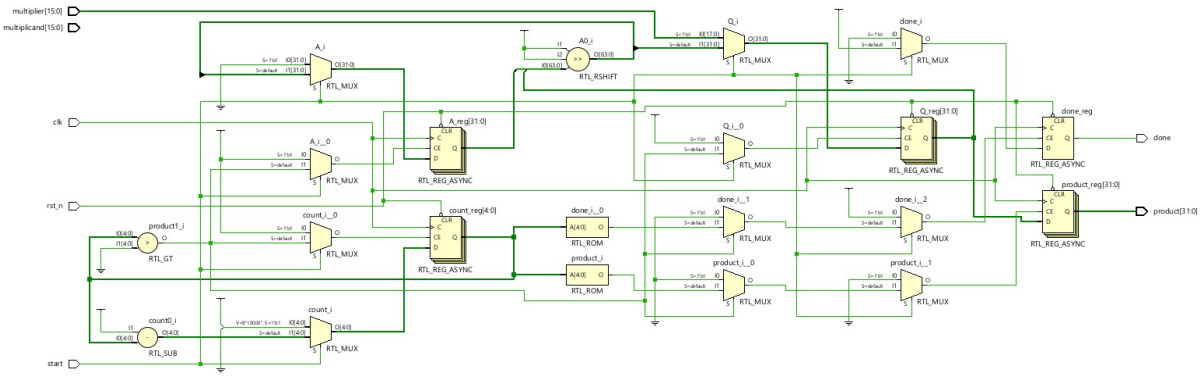

Figure 2: Schematic of Booth Multiplier

9 Synthesis Design

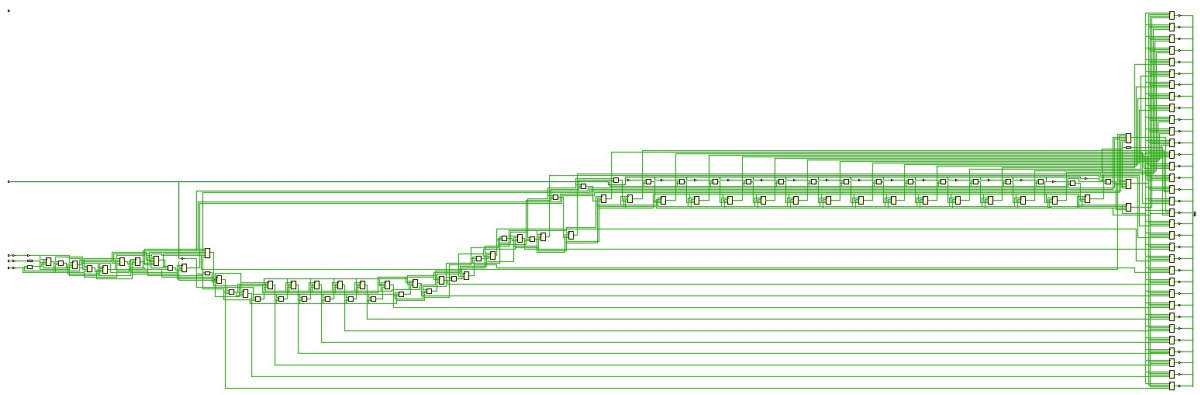

Figure 3: Synthesis of Booth Multiplier

10 Conclusion

The Booth Multiplier is a notable advancement in the field of binary multiplication, particularly for signed number operations. By employing Booth's algorithm, the multiplier efficiently reduces the number of addition and subtraction operations through smart encoding of the multiplier. This approach significantly improves performance, especially for numbers with long sequences of 1's, and makes the Booth Multiplier an effective solution for handling two's complement signed multiplication.

The advantages of the Booth Multiplier include fewer operations, efficient handling of signed numbers, and reduced hardware complexity in comparison to traditional multipliers. Its scalability and flexibility in processing large bit-widths also make it a suitable choice for high-performance applications. However, it does come with some challenges, such as increased control complexity, variable operation times, and potential power consumption concerns.

In conclusion, the Booth Multiplier is a powerful and efficient tool for signed binary multiplication, offering improved performance in arithmetic operations where speed and efficiency are critical. Its applications span across various domains such as digital signal processing, cryptography, and embedded systems, making it a valuable component in modern digital designs. As computational demands continue to rise, the Booth Multiplier remains a key architecture for optimizing signed multiplication in digital circuits.

11 Frequently Asked Questions (FAQs)

11.1 What is a Booth Multiplier?

A Booth Multiplier is a digital circuit that uses Booth's algorithm to efficiently multiply binary numbers, particularly signed numbers, by reducing the number of addition and subtraction operations through smart encoding of the multiplier.

11.2 How does the Booth Multiplier improve multiplication efficiency?

The Booth Multiplier improves efficiency by encoding consecutive 1's in the multiplier, allowing for fewer addition and subtraction operations. This reduces the overall number of operations, making the

multiplication process faster, especially for large bit-width numbers.

11.3 What are the main components of a Booth Multiplier?

The main components of a Booth Multiplier include input ports for the multiplicands, a Booth Encoder for generating control signals, an adder/subtractor unit for computing partial products, a shift register for alignment, and output ports for the final product.

11.4 In what applications is the Booth Multiplier commonly used?

The Booth Multiplier is widely used in applications requiring signed binary multiplication, such as digital signal processing, cryptography, embedded systems, and real-time computing environments.

11.5 What are the trade-offs of using a Booth Multiplier?

While the Booth Multiplier reduces the number of operations and handles signed numbers efficiently, it introduces complexity in control logic, may have variable operation times depending on the bit pattern of the multiplier, and can lead to higher power consumption.

11.6 Can the Booth Multiplier handle large bit-widths?

Yes, the Booth Multiplier is scalable and capable of handling large bit-width multiplications without significantly increasing complexity, making it suitable for various high-performance digital systems.