Project 35: Sign Radix-4 Modified booth Multiplier

A Comprehensive Study of Advanced Digital Circuits

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

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Project Overview

The Sign Radix-4 Modified Booth Multiplier is a digital multiplier used in computer architecture for efficient multiplication of signed numbers. It employs a modified version of Booth's algorithm, utilizing radix-4 encoding to reduce the number of partial products generated during multiplication. This method groups bits into pairs, allowing for fewer additions and shifts compared to conventional methods. It improves speed and reduces complexity, making it suitable for high-performance computing applications where efficient arithmetic operations are essential.

2 Sign Radix-4 Modified booth Multiplier

2.1 Basic Concept of Sign Radix-4 Modified booth Multiplier

The Sign Radix-4 Modified Booth Multiplier efficiently multiplies signed integers using a modified Booth's algorithm. It encodes the multiplier into pairs of bits, reducing the number of partial products generated. Each pair is evaluated as follows:

- **00:** *No addition.*
- **01:** *Add the multiplicand.*
- **10:** *Subtract the multiplicand.*
- **11:** *Add twice the multiplicand.*

This method allows for a shift-and-add process that accumulates results while effectively handling signs, improving speed and reducing complexity in digital multiplication.

2.2 Architecture of Sign Radix-4 Modified booth Multiplier

- **Inputs:** Accepts the multiplicand and multiplier.
- **Booth Encoder:** Encodes the multiplier into pairs of bits for control signals.
- **Partial Product Generation:** Creates partial products based on encoded signals (add, subtract, or no operation).
- **Shift Register:** Shifts the accumulated result left during the process.
- **Adder/Subtractor:** Combines partial products with the accumulated result.
- **Accumulator:** Stores the ongoing sum of partial products.
- **Control Logic:** Manages the overall operation of the multiplier.
- **Output:** Produces the final product.

This architecture optimizes speed and efficiency in signed multiplication.

2.3 Working of Sign Radix-4 Modified booth Multiplier

The working of the Sign Radix-4 Modified Booth Multiplier involves several steps to perform signed multiplication efficiently. Here's a concise overview of the process:

- **Initialization:** Load the multiplicand (M) and multiplier (Q) into the system. Initialize an accumulator (A) to zero to hold the result.
- **Booth Encoding:** Examine the last two bits of the multiplier (Q) and an additional bit (Q-1) initialized to 0. Based on the pairs (Q1, Q0, Q-1), determine the action:
    1. *00: No action.*

2. *01: Add the multiplicand (M) to the accumulator (A).*
3. *10: Subtract the multiplicand (M) from the accumulator (A).*
4. *11: Add double the multiplicand (2M) to the accumulator (A).*

- **Partial Product Generation:** Perform the appropriate operation determined by the Booth encoding.
- **Shifting:** Shift the accumulator (A) and multiplier (Q) right by one bit. The sign bit of the accumulator is preserved to maintain the sign of the result.
- **Iteration:** Repeat the encoding, partial product generation, and shifting steps for each bit of the multiplier, typically until all bits have been processed (equal to the number of bits in the multiplier).
- **Final Result:** The final product is obtained in the accumulator (A) after processing all bits. The result may be in two's complement form if both numbers are signed.

2.4 RTL Code

Listing 1: Sign Radix-4 Modified booth Multiplier

```
1
2  module radix4_booth_multiplier (
3      input  logic signed [7:0] A,  // 8-bit signed input A
4      input  logic signed [7:0] B,  // 8-bit signed input B
5      output logic signed [15:0] product  // 16-bit signed product
6  );
7      // Booth encoding and partial product generation logic here
8      logic signed [15:0] pp[0:3];
9      logic [1:0] booth;
10
11     always_comb begin
12         product = 0;
13         for (int i = 0; i < 4; i++) begin
14             booth = {B[2*i+1], B[2*i]};
15             case (booth)
16                 2'b00, 2'b11: pp[i] = 0;            // 0 times A
17                 2'b01:        pp[i] = A << (2*i);   // +A shifted
18                 2'b10:        pp[i] = -(A << (2*i)); // -A shifted
19             endcase
20             product = product + pp[i];
21         end
22     end
23 endmodule
```

2.5 Testbench

Listing 2: Sign Radix-4 Modified booth Multiplier

```
1
2  module tb_radix4_booth_multiplier;
3      logic signed [7:0] A, B;
4      logic signed [15:0] product;
5
6      radix4_booth_multiplier uut (.A(A), .B(B), .product(product));
7
8      initial begin
9          // Test case 1
10         A = 8'sb00011010; // 26
11         B = 8'sb00000101; // 5
12         #10;
13         $display("A = %d, B = %d, Product = %d", A, B, product);
14
15         // Test case 2
16         A = 8'sb11111111; // -1
17         B = 8'sb00000010; // 2
18         #10;
19         $display("A = %d, B = %d, Product = %d", A, B, product);
20
21         $finish;
22     end
23 endmodule
```


3 Results

3.1 Simulation

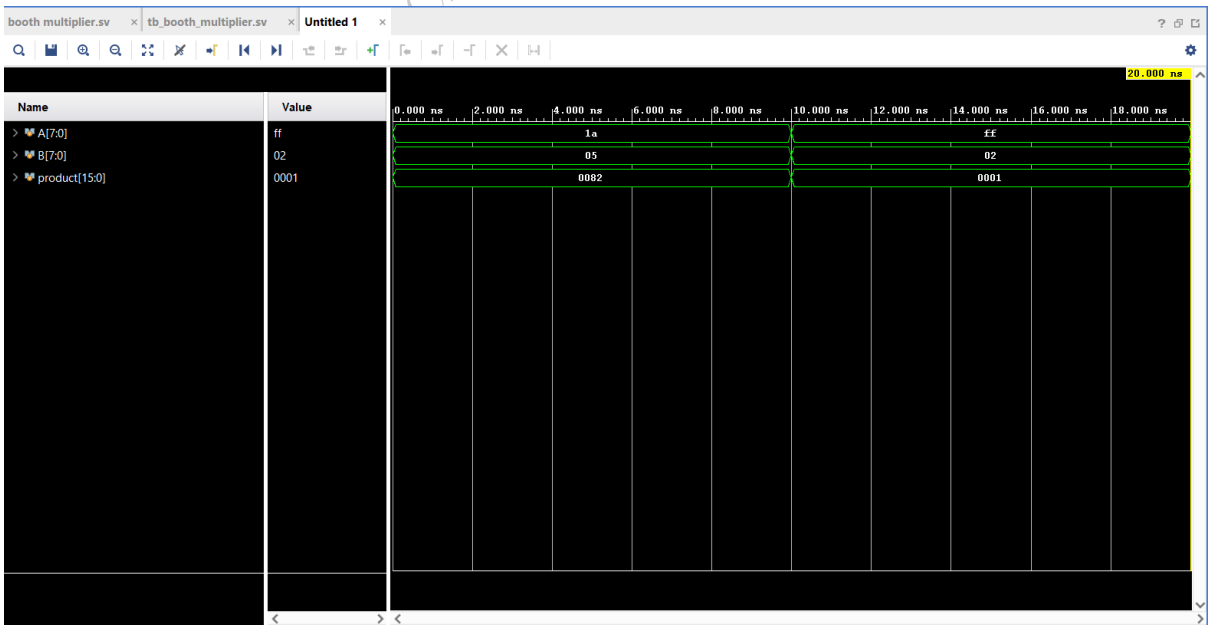

Figure 1: Simulation of Sign Radix-4 Modified booth Multiplier

3.2 Schematic

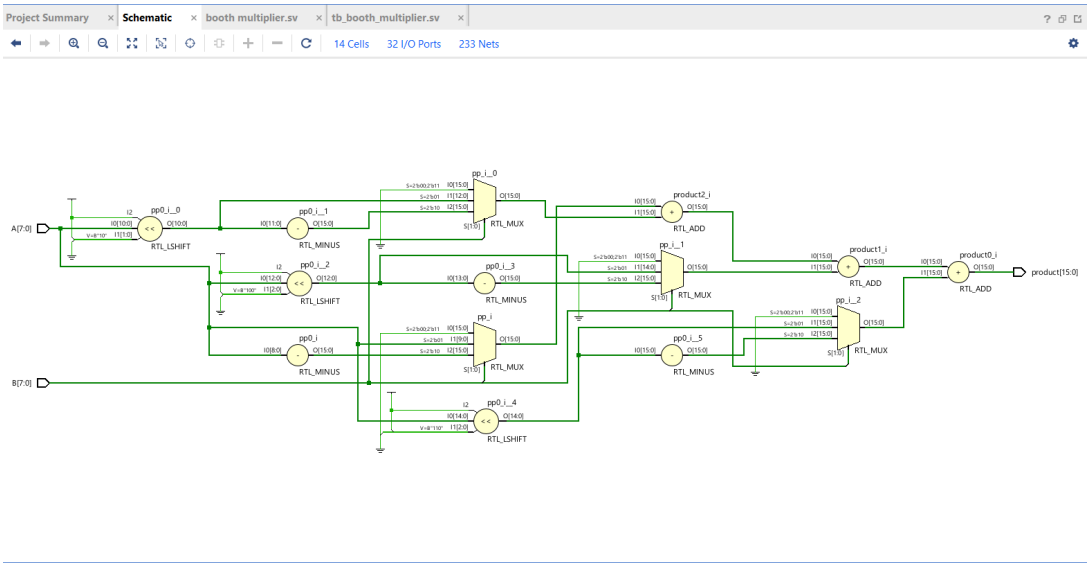

Figure 2: Schematic of Sign Radix-4 Modified booth Multiplier

3.3 Synthesis Design

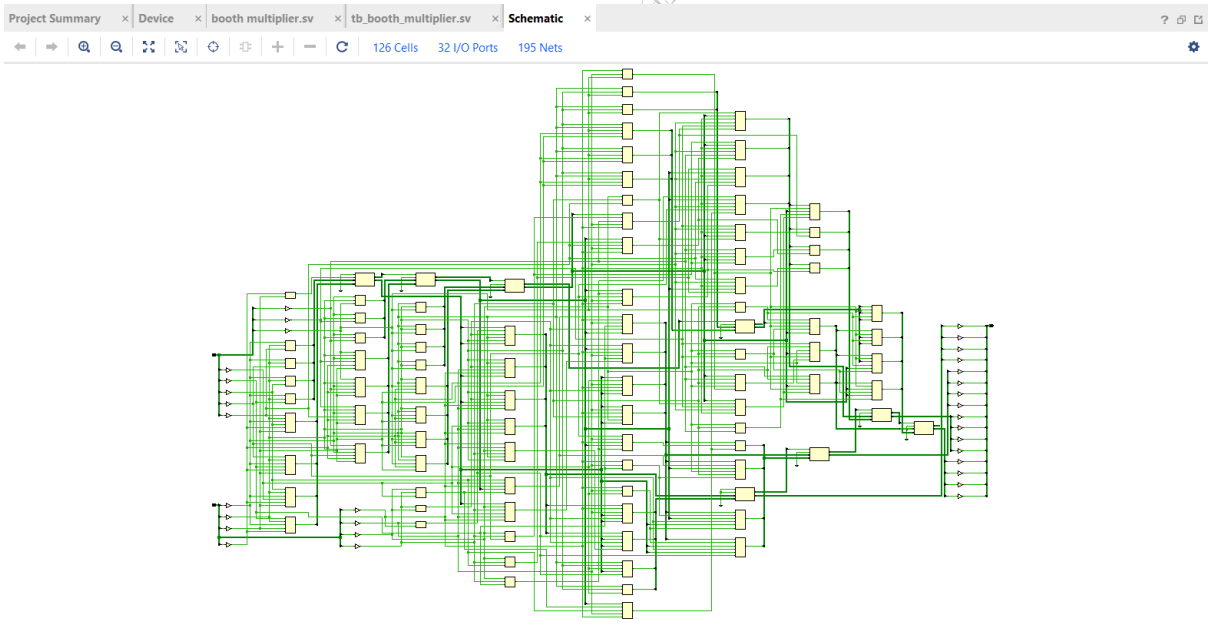

Figure 3: Synthesis Design of Sign Radix-4 Modified booth Multiplier

4 Advantages of Sign Radix-4 Modified booth Multiplier

- **Reduced Partial Products:** By processing two bits at a time, it generates fewer partial products compared to traditional methods, minimizing the overall number of additions and shifts required.
- **Speed:** The reduced number of operations leads to faster multiplication, making it suitable for high-performance applications, such as digital signal processing and graphics.
- **Efficiency in Signed Multiplication:** It effectively handles both positive and negative numbers using two's complement representation, streamlining the multiplication of signed integers.
- **Lower Complexity:** The architecture requires fewer hardware resources, which can simplify the design and integration into digital circuits, such as FPGAs or ASICs.
- **Better Utilization of Hardware:** The shift-and-add approach can take advantage of parallelism in hardware implementations, further enhancing performance.
- **Scalability:** The design can be easily scaled for larger bit-width multiplications, making it versatile for various applications in computer architecture.
- **Reduced Power Consumption:** Fewer operations and simpler circuitry can lead to lower power consumption, which is crucial for battery-operated and portable devices.

5 Disadvantages of Sign Radix-4 Modified booth Multiplier

- **Complexity of Control Logic:** The control logic needed to manage the encoding and decoding process can become complex, potentially increasing design time and effort.
- **Latency in Processing:** While it reduces the number of partial products, the shifting and addition operations can introduce latency, especially in high-speed applications where quick response times are critical.
- **Limited Precision:** The architecture may be less precise in handling very large numbers or certain edge cases, which can lead to overflow or underflow issues.
- **Hardware Resource Requirements:** Although it reduces the number of partial products, the overall hardware complexity can still be higher compared to simpler multiplication methods, especially in terms of the adder/subtractor units.
- **Implementation Challenges:** Designing and implementing the multiplier in hardware can be challenging, particularly when integrating with existing architectures.
- **Dependency on Bit-width:** The efficiency gains may be less significant for smaller bit-width multiplications, where simpler multipliers may perform adequately without the overhead of Booth's algorithm.
- **Potential for Increased Area:** In some cases, the use of additional components for encoding and the complexity of the adder structures can lead to increased silicon area in integrated circuits.
- **Loss of Precision:**Truncation introduces a small error in the final result, as the least significant partial products are ignored or approximated. This may be unacceptable for applications requiring high precision.
- **Error Management:**The error introduced by truncation needs to be managed carefully, especially in sensitive applications. This may require additional error compensation techniques, which could increase the design complexity.

- **Limited Use in Precision-Critical Applications:**Truncated multipliers are not suitable for applications that demand exact or high-precision results, such as scientific computing or cryptography.
- **Design Complexity in Compensation:**Implementing error compensation methods (such as constant bias addition or adaptive correction) adds complexity to the design, potentially offsetting some of the benefits in terms of simplicity and power savings.
- **Non-Uniform Error Distribution:**The error introduced by truncation is often non-uniform across different input combinations, leading to variable levels of accuracy depending on the specific values being multiplied.

6 Applications of Sign Radix-4 Modified booth Multiplier

- **Digital Signal Processing (DSP):**Commonly used in DSP systems for fast multiplication operations, such as filtering and convolution, where efficiency is crucial.
- **Image Processing:**Employed in image processing algorithms for operations like scaling, rotation, and color conversion, which require frequent multiplications.
- **Computer Graphics:**Utilized in graphics rendering and 3D transformations, where fast arithmetic calculations are essential for real-time performance.
- **Embedded Systems:**Implemented in microcontrollers and FPGAs for applications that require efficient arithmetic operations, including control systems and robotics.
- **Machine Learning and AI:**Used in neural networks and other machine learning algorithms where numerous multiplications are performed, especially in training models.
- **Signal Modulation/Demodulation:**Applied in communication systems for tasks such as modulation and demodulation, where multiplying signals is fundamental.
- **Cryptography:**Utilized in cryptographic algorithms that require secure and efficient multiplications, such as those used in public-key cryptography.
- **Arithmetic Logic Units (ALUs):**Integrated into the arithmetic units of CPUs and GPUs to enhance their computational capabilities for arithmetic operations.

7 Conclusion

In conclusion, the Sign Radix-4 Modified Booth Multiplier is a powerful and efficient solution for performing signed multiplication in digital systems. Its advantages, including reduced partial products, improved speed, and lower complexity, make it particularly well-suited for high-performance applications in digital signal processing, computer graphics, image processing, and machine learning. While it does present some challenges in terms of control logic complexity and hardware resource requirements, its benefits often outweigh these drawbacks in scenarios where efficient arithmetic operations are critical. As technology continues to evolve, the Sign Radix-4 Modified Booth Multiplier remains a valuable component in modern computing architectures, enabling faster and more efficient processing in a variety of applications.

8 FAQs

1. What is the main advantage of using the Sign Radix-4 Modified Booth Multiplier over traditional multiplication methods?

The main advantage is its ability to reduce the number of partial products generated, leading to fewer additions and shifts. This results in faster computation times and lower hardware complexity, especially

in signed multiplication.

2. How does the Radix-4 encoding work in the Booth multiplier?
Radix-4 encoding processes two bits of the multiplier at a time, allowing for four possible combinations (00, 01, 10, 11). Each combination dictates whether to add, subtract, or perform no operation with the multiplicand, thereby reducing the overall number of partial products.

3. Is the Sign Radix-4 Modified Booth Multiplier suitable for both signed and unsigned numbers?
Yes, it is primarily designed for signed numbers, utilizing two's complement representation. However, it can also handle unsigned numbers effectively by treating them as positive values.

4. What is the significance of the shift-and-add process in the multiplier?
The shift-and-add process allows for the accumulation of results while preserving the sign of the accumulated value. It helps in efficiently combining the generated partial products to obtain the final product.

5. What types of applications benefit from the Sign Radix-4 Modified Booth Multiplier?
Applications in digital signal processing, computer graphics, image processing, machine learning, cryptography, and embedded systems benefit from its efficiency and speed in performing arithmetic operations.

6. Are there any disadvantages to using the Sign Radix-4 Modified Booth Multiplier?
Yes, disadvantages include increased complexity in control logic, potential latency in processing, and higher hardware resource requirements compared to simpler multiplication methods. Implementation can also be challenging in some scenarios.

7. Can the Sign Radix-4 Modified Booth Multiplier be implemented in hardware?
Absolutely. It is commonly implemented in FPGAs and ASICs, where its efficient arithmetic capabilities can significantly enhance the performance of various digital systems.

8. How does the Sign Radix-4 Modified Booth Multiplier handle overflow or underflow?
The multiplier uses two's complement representation for signed numbers, so overflow or underflow issues can occur when the final result exceeds the representable range of the data type. It is essential to implement checks to handle such conditions in practical applications.

9. How does the performance of the Sign Radix-4 Modified Booth Multiplier compare to other multipliers?
The Sign Radix-4 Modified Booth Multiplier typically outperforms traditional multipliers in terms of speed and resource efficiency, especially in applications that involve frequent signed multiplication operations. However, the actual performance may vary based on specific implementation and hardware.

10. Is there a limit to the bit-width for which the Sign Radix-4 Modified Booth Multiplier can be effectively used?
While there is no strict limit, the complexity and resource requirements increase with larger bit-widths. The efficiency gains may be less significant for smaller bit-width multiplications, where simpler methods may suffice.