ECE 310
Project 1: 8×8 Wallace Tree Multiplier

Reewaj Adhikari

September 18, 2025

Abstract

This project presents the design and implementation of an 8×8 Wallace tree multiplier using structural Verilog modeling. The Wallace multiplier algorithm was implemented through a multi-stage reduction tree that combines partial products using half adders and full adders to minimize propagation delay. The design consists of partial product generation using 64 AND gates, five levels of carry-save addition reduction using 17 half adders and 47 full adders, and final product output assignment. A comprehensive testbench verified the multiplier's functionality across various input combinations. The implementation successfully demonstrated the Wallace tree's ability to perform high-speed multiplication through parallel processing of carry-save additions.

The Wallace tree multiplier is a high-speed digital multiplier that reduces sequential addition steps by using a tree of carry-save adders instead of ripple-carry adders. It operates in three phases:

1. **Partial Product Generation:** 64 AND gates produce $p[i][j] = a[i] \cdot b[j]$
2. **Reduction Tree:** Half and full adders reduce 8 rows of partial products to 2
3. **Final Addition:** The remaining 2 rows produce the 16-bit product

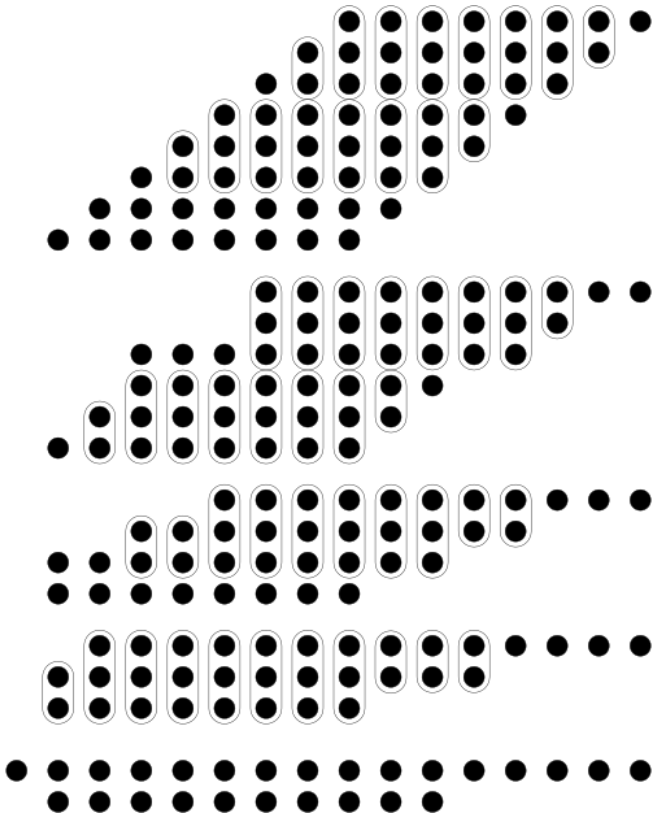

Figure 1: Wallace Tree Diagram.

1 Design and Implementation

1.1 Basic Building Blocks

The design utilizes two fundamental components: half adders and full adders implemented at the gate level.

Listing 1: Half Adder Implementation

```
module half_adder(
    output S, Cout,
    input A, B
);
    xor(S, A, B);
    and (Cout, A, B);
endmodule
```

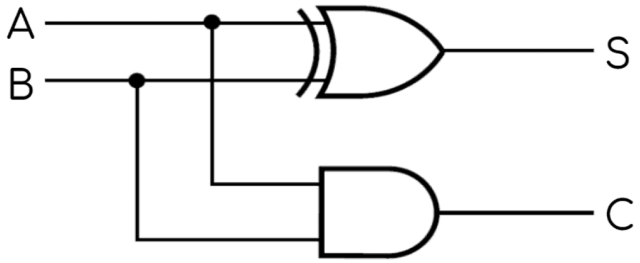

Figure 2: Half Adder Diagram.

Listing 2: Full Adder Implementation

```
module full_adder (
    output S, Cout,
    input A, B, Cin
);
    wire w1, w2, w3;
    xor (w1, A, B);
    xor (S, w1, Cin);
    and (w2, A, B);
    and (w3, w1, Cin);
    or (Cout, w2, w3);
endmodule
```

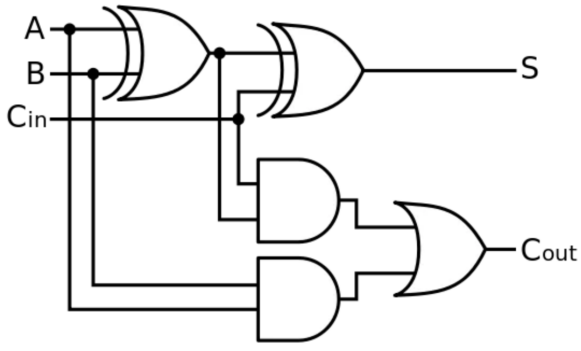

Figure 3: Full Adder Diagram.

1.2 Partial Product Generation

The 8×8 Wallace tree multiplier generates all 64 partial products simultaneously using AND gates:

Listing 3: Partial Product Generation

```
wire p[7:0][7:0];
genvar i, j;
generate
for (i = 0; i < 8; i = i + 1) begin : gen_i
    for (j = 0; j < 8; j = j + 1) begin : gen_j
        and and_gates ((p[i][j]),(a[i]),(b[j]));
    end
end
endgenerate
```

This creates a matrix where each element $p[i][j]$ represents the product of bit i from operand a and bit j from operand b.

1.3 Complete Wallace Tree Structure

The complete 8×8 Wallace multiplier structure shows the systematic arrangement of partial products and their reduction through multiple levels of half adders and full adders:

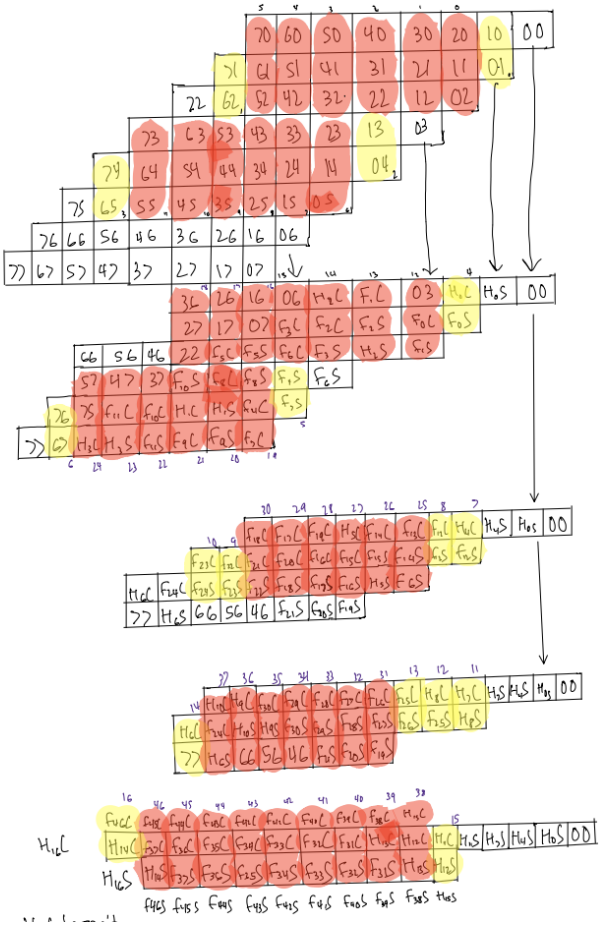

Figure 4: Full Wallace Multiplier Diagram.

The Wallace tree reduction is implemented through five levels of adders, with each level strategically combining partial products to minimize the total number of levels required. The tree structure allows maximum parallelism while ensuring all carries are properly propagated, ultimately reducing the partial product matrix from 8 rows down to 2 rows for final addition.

1.4 Testbench Design

A comprehensive testbench was developed to verify the multiplier's functionality across various scenarios:

Listing 4: Wallace Multiplier Testbench

```
initial begin
    // Test 1: Zero case
    a = 8'd0; b = 8'd0;
    // Test 2: Identity multiplication
    a = 8'd1; b = 8'd1;
    // Test 3: Small multiplications
    a = 8'd3; b = 8'd5;
    // Test 4: Medium values
    a = 8'd15; b = 8'd15;
    // Test 5: High bit set case
    a = 8'd128; b = 8'd5;
    // Test 6: Large values
    a = 8'd100; b = 8'd200;
    // Test 7: Maximum case
    a = 8'd255; b = 8'd255;
    $finish;
end
```

2 Results and Analysis

The simulation results confirmed the correct functionality of the Wallace tree multiplier for all test cases. The testbench output demonstrated perfect accuracy across all seven test vectors:

- **Zero multiplication:** $0 \times 0 = 0$ verified baseline functionality
- **Identity multiplication:** $1 \times 1 = 1$ confirmed basic operation
- **Small values:** $3 \times 5 = 15$ tested simple multiplication
- **Medium values:** $15 \times 15 = 225$ verified carry propagation through multiple levels
- **Large operands:** $128 \times 5 = 640$ tested MSB handling and asymmetric inputs
- **High-result case:** $100 \times 200 = 20000$ confirmed large product generation without overflow
- **Maximum case:** $255 \times 255 = 65025$ verified full-scale operation using all 16 output bits

Each test case produced the exact expected result, confirming that the Wallace multiplier successfully computed all products within the 16-bit result range. The structural implementation demonstrated proper partial product generation, systematic carry-save reduction across five levels, and accurate final product assembly.

The most critical test was the maximum case (255×255), which exercises all 64 partial products simultaneously and requires maximum carry propagation through the entire reduction tree. The correct result of 65025 validates the complete functionality of the multiplier under maximum load conditions.

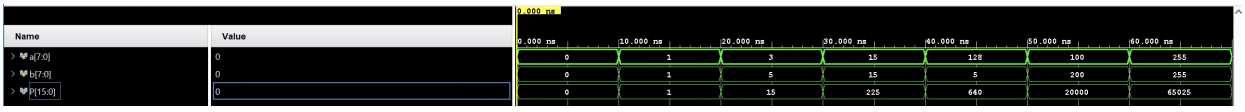

Figure 5: Waveform Diagram.

3 Component Analysis

The complete Wallace multiplier implementation utilizes:

- **64 AND gates** for partial product generation
- **17 half adders** for 2-input reductions
- **47 full adders** for 3-input reductions
- **16 buffer gates** for output assignment

The choice between half adders and full adders was made systematically based on the number of operands at each bit position in the reduction tree. Half adders were used when only two signals needed to be added, while full adders were employed for three-signal additions, optimizing both area and performance.

4 Performance Analysis

The Wallace tree multiplier is faster than regular multipliers because it adds many numbers at the same time instead of one after another. This means it can multiply numbers more quickly.

Some key points about its performance:

- **Faster Multiplication:** The tree structure reduces the time it takes to get the final result.
- **Multiple Additions at Once:** Many partial products are added in parallel, which speeds up the process.
- **Works for Bigger Numbers:** The method can be used for larger numbers without much extra delay.

The tradeoff is that it uses more hardware (lots of adders) to achieve this speed. In this design, 144 components are used, which makes it faster but takes up more space compared to simpler multipliers.

5 Conclusion

The 8×8 Wallace tree multiplier was successfully implemented using structural Verilog modeling with complete gate-level design. The implementation demonstrated the effectiveness of the Wallace algorithm in achieving high-speed multiplication through parallel carry-save addition. All seven test cases passed verification, confirming proper functionality across the full input range from zero to maximum values.

The project highlighted both the performance benefits and hardware complexity of the Wallace multiplier architecture. The systematic reduction tree approach, optimal use of half and full adders, and comprehensive testing methodology resulted in a robust multiplier design well-suited for applications requiring fast multiplication operations where hardware resources are available.

The modular design approach using gate-level building blocks demonstrated sound digital design principles and facilitated systematic debugging and verification. The comprehensive testbench provided high confidence in the design's correctness and robustness across diverse operating conditions.

This report was compiled using LaTeX.