Project 12: Ling Adder

A Comprehensive Study of Advanced Digital Circuits

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

Contents

1 Introduction

The Ling Adder is an advanced variation of the carry-lookahead adder, designed to enhance the speed of binary addition while minimizing hardware complexity. It achieves this by efficiently generating intermediate signals that simplify the computation of carry bits. The Ling Adder is known for its ability to optimize the critical path delay, making it a valuable component in high-speed arithmetic circuits and performance-sensitive applications.

2 Key Concepts

1. **Intermediate Signal Generation**: The Ling Adder creates intermediate signals to simplify carry computation, leading to faster addition.
2. **Efficient Carry Propagation**: It uses generate (g) and propagate (p) signals to optimize carry propagation, reducing the critical path delay.
3. **Optimized Area and Delay**: By reducing the number of logic gates, the Ling Adder achieves a balance between speed and resource usage, making it ideal for high-performance systems.

3 Steps in Ling Addition

1. **Generate and Propagate Signals:**
   - Calculate the generate signal $G_i = A_i \cdot B_i$.
   - Calculate the propagate signal $P_i = A_i + B_i$.
2. **Compute Ling Propagate Signal:**
   - Modify the propagate signal:
     $$P_i' = P_i \cdot P_{i-1}$$
     to simplify carry computation.
3. **Calculate Carry Signals:**
   - Compute the carry recursively:
     $$C_{i+1} = G_i + (P_i' \cdot C_i)$$
     reducing carry computation complexity.
4. **Sum Calculation:**
   - Compute the sum:
     $$S_i = A_i \oplus B_i \oplus C_i$$
5. **Output:**
   - The final result is the sum S with a possible carry-out.

4 Diagrams

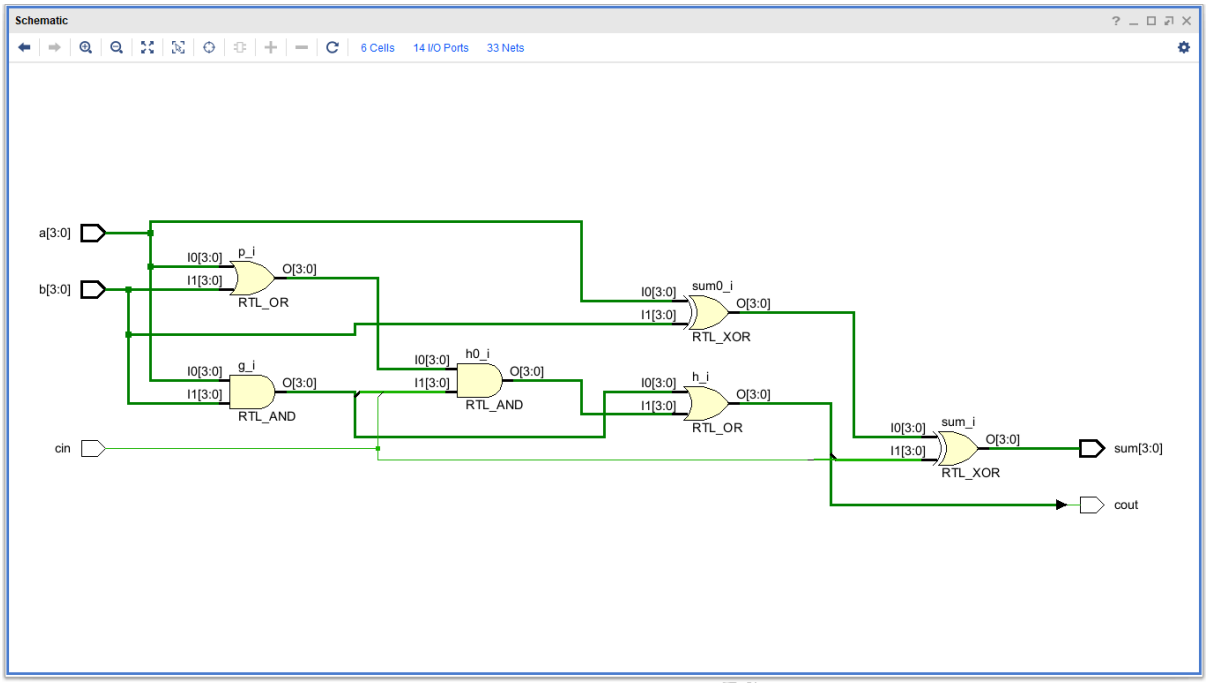

Figure 1: Schematic of Ling Adder

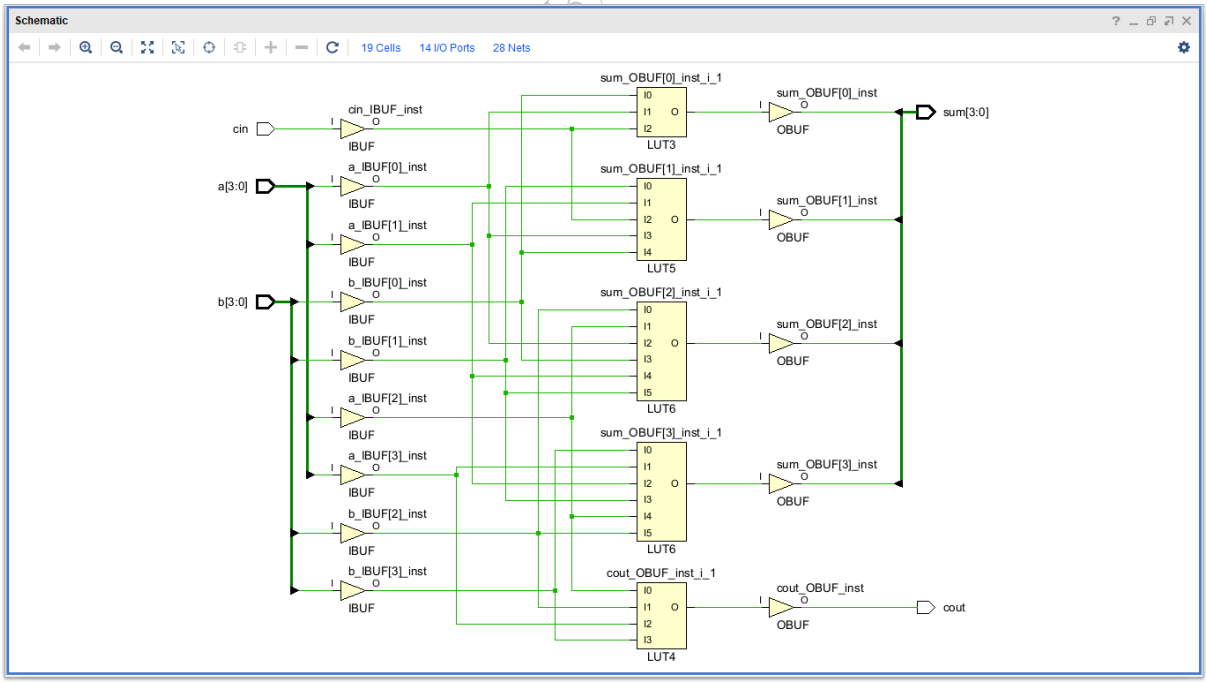

Figure 2: Synthesis of Ling Adder

5 Why to Choose It

The Ling Adder is chosen for its efficiency in reducing the carry propagation delay. By simplifying the carry generation logic compared to traditional adders, it achieves faster addition with reduced circuit complexity. This makes it an excellent choice for high-performance arithmetic units where speed is crucial, particularly in applications requiring rapid data processing with minimal power consumption.

6 SystemVerilog Code

Listing 1: Ling Adder RTL Code

```
module ling_adder #(parameter WIDTH = 4) (
    input logic [WIDTH-1:0] a,
    input logic [WIDTH-1:0] b,
    input logic cin,
    output logic [WIDTH-1:0] sum,
    output logic cout
);

    logic [WIDTH-1:0] g;  // Generate
    logic [WIDTH-1:0] p;  // Propagate
    logic [WIDTH-1:0] h;  // Ling's carry
    logic [WIDTH-1:0] c;  // Carry

    // Generate and Propagate
    assign g = a & b;
    assign p = a  b;

    // Ling's carry computation
    assign h = g  (p & {g[WIDTH-2:0], cin});

    // Carry computation
    assign c[0] = cin;
    assign c[WIDTH-1:1] = h[WIDTH-2:0];

    // Sum computation
    assign sum = a ^ b ^ c;
    assign cout = h[WIDTH-1];

endmodule
```

7 Testbench

Listing 2: Ling Adder Testbench

```
module tb_ling_adder;

    parameter WIDTH = 4;

    logic [WIDTH-1:0] a;
    logic [WIDTH-1:0] b;
    logic cin;
    logic [WIDTH-1:0] sum;
    logic cout;

```

```
    // Instantiate the Ling Adder
    ling_adder #(.WIDTH(WIDTH)) uut (
        .a(a),
        .b(b),
        .cin(cin),
        .sum(sum),
        .cout(cout)
    );

    // Testbench stimulus
    initial begin
        // Apply test vectors
        a = 4'b0000; b = 4'b0000; cin = 0; #10;
        $display("Test Case 1: a=%b, b=%b, cin=%b, sum=%b, cout=%b",
            a, b, cin, sum, cout);

        a = 4'b0011; b = 4'b0101; cin = 0; #10;
        $display("Test Case 2: a=%b, b=%b, cin=%b, sum=%b, cout=%b",
            a, b, cin, sum, cout);

        a = 4'b1111; b = 4'b0001; cin = 1; #10;
        $display("Test Case 3: a=%b, b=%b, cin=%b, sum=%b, cout=%b",
            a, b, cin, sum, cout);

        a = 4'b1010; b = 4'b1100; cin = 0; #10;
        $display("Test Case 4: a=%b, b=%b, cin=%b, sum=%b, cout=%b",
            a, b, cin, sum, cout);

        a = 4'b0110; b = 4'b0011; cin = 1; #10;
        $display("Test Case 5: a=%b, b=%b, cin=%b, sum=%b, cout=%b",
            a, b, cin, sum, cout);

        $stop;
    end

endmodule
```

8 How It Works In-Depth

The Ling Adder operates through a series of steps:

8.1 Initialization

Generate (G) and propagate (P) signals are computed for each bit:

$$G_i = A_i \cdot B_i$$
$$P_i = A_i + B_i$$

8.2 Modified Propagate Signal Computation

The modified propagate signal (P_i') is computed to simplify carry computation:

$$P_i' = P_i \cdot P_{i-1}$$

This step reduces the complexity of carry calculations by adjusting the propagate signal.

8.3 Carry Computation

Carry signals are computed using the generate signals and modified propagate signals:

$$C_{i+1} = G_i + (P_i' \cdot C_i)$$

This recursive formula allows efficient carry propagation across multiple bits.

8.4 Sum Computation

Final sum bits are computed using:

$$S_i = A_i \oplus B_i \oplus C_i$$

where C_i is the carry for the current bit position.

8.5 Simulation Setup

Testbench Configuration:

- The testbench module instantiates the Ling adder and includes stimulus generation for various test cases.
- It applies a range of binary inputs to verify the adder's functionality.

Clock Frequency and Simulation Duration:

- Clock frequency: [Specify the clock frequency, e.g., 100 MHz].
- Simulation duration: [Specify the duration, e.g., 100 ns].

Inputs and Expected Outputs:

- **Basic Functionality Tests:**
  - Example input: $A = 1010$, $B = 0101$
  - Expected output: $S = 1111$, Carry-out $= 0$
- **Edge Cases:**
  - Example input: $A = 1111$, $B = 1111$
  - Expected output: $S = 1110$, Carry-out $= 1$
- **Overflow Conditions:**
  - Example input: $A = 1111$, $B = 0001$
  - Expected output: $S = 0000$, Carry-out $= 1$

8.6 Simulation Results

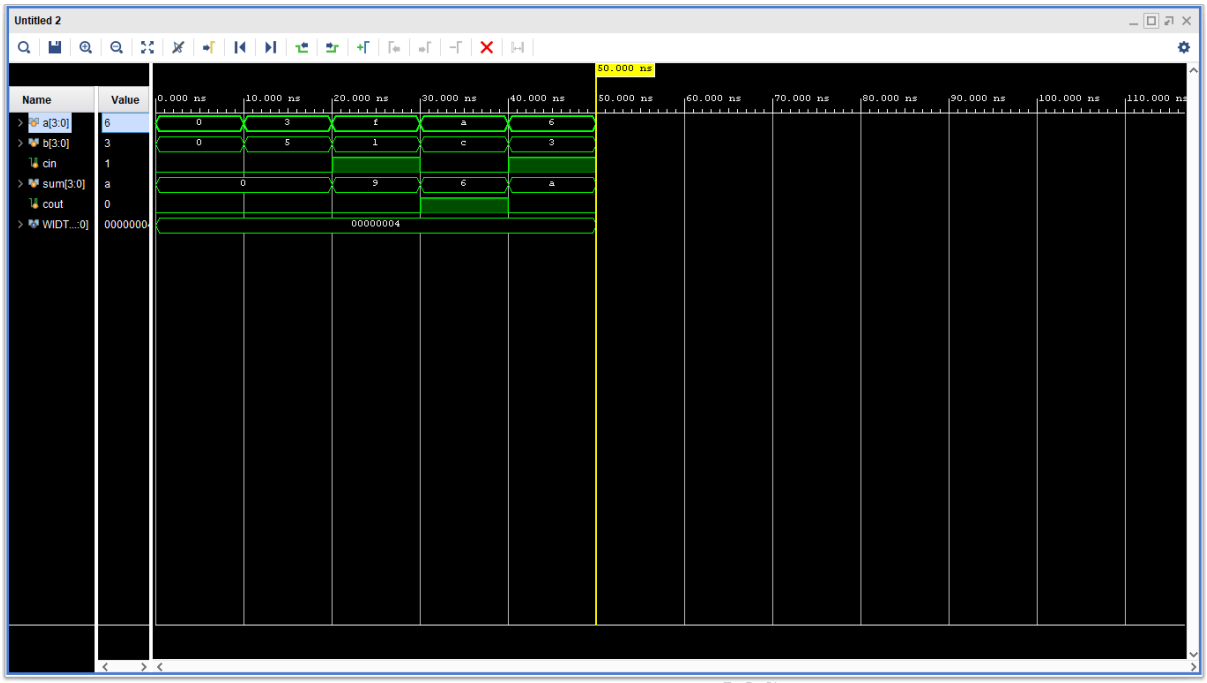

Figure 3: Simulation of Ling Adder

9 Conclusion

The Ling adder implementation in Vivado demonstrates the efficiency and scalability of this advanced adder design, particularly for high-performance computing applications. By optimizing the carry computation through the use of modified propagate signals, the Ling adder offers faster addition with reduced delay compared to traditional adder designs.

10 References

1. H. T. Kung, C. E. Leiserson, and M. L. Sicherman, "A New Efficient Algorithm for the Addition of Two Binary Numbers," *IEEE Transactions on Computers*, vol. C-23, no. 11, pp. 1208-1213, Nov. 1974.

2. D. P. Harris, "A Family of Fast Adders," *IEEE Transactions on Computers*, vol. C-24, no. 5, pp. 464-473, May 1975.

3. J. F. Wakerly, *Digital Design: Principles and Practices*, 4th ed., Pearson, 2014.

4. S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed., McGraw-Hill Education, 2013.