Project 18: Logarithmic Adder

A Comprehensive Study of Advanced Digital Circuits

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

Contents

1 Introduction 3

2 Background 3

3 Structure and Operation 3

4 Implementation in SystemVerilog 4

5 Test Bench 4

6 Advantages and Disadvantages 5
- 6.1 Advantages . . . 5
- 6.2 Disadvantages . . . 5

7 Conclusion 5

8 Simulation Results 6

9 Schematic 6

10 Synthesis Design 6

1 Introduction

The **Logarithmic Adder** is an advanced digital circuit designed to perform addition operations using a logarithmic approach. Unlike traditional adders that use linear arithmetic operations, the logarithmic adder utilizes logarithmic functions to accelerate the addition process, thereby improving computational efficiency. This approach is particularly beneficial in high-speed and high-precision applications where traditional adders might struggle with performance.

2 Background

Traditional adders, such as the Ripple Carry Adder (RCA) and the Carry Look-Ahead Adder (CLA), rely on linear arithmetic operations and carry propagation mechanisms. While these adders are effective for general purposes, they may encounter limitations in speed and efficiency as bit-width increases.

The Logarithmic Adder addresses these limitations by leveraging the properties of logarithms. In essence, the logarithmic adder converts the addition operation into a multiplication operation using logarithms, which can be performed more rapidly. The basic principle involves:

- Converting the operands into their logarithmic forms.
- Performing multiplication in the logarithmic domain.
- Converting the result back to the original domain.

This method can significantly reduce propagation delays and improve performance in complex arithmetic operations.

3 Structure and Operation

The Logarithmic Adder comprises several key components:

- **Logarithmic Conversion Units**: Convert the input operands into their logarithmic representations.
- **Multiplication Unit**: Performs multiplication in the logarithmic domain.
- **Inverse Logarithmic Conversion Unit**: Converts the result back to the linear domain.
- **Output Port**: Provides the final result of the addition operation.

The operation of the Logarithmic Adder can be summarized as follows:

1. Convert inputs A and B into logarithmic form $\log(A)$ and $\log(B)$.
2. Multiply the logarithmic values to obtain $\log(A) + \log(B)$.
3. Convert the result back to the linear domain to obtain $A + B$.
4. Output the result.

Design Considerations When designing the Logarithmic Adder, the following factors must be considered:

- *Precision*: Ensure accurate conversion and computation to avoid errors due to logarithmic approximations. - *Speed*: Optimize the multiplication and conversion units to achieve high-speed performance. - *Hardware Resources*: Consider the trade-off between hardware complexity and performance improvements. - *Power Consumption*: Minimize power usage while maintaining high performance.

Performance Metrics The performance of the Logarithmic Adder can be evaluated based on several key metrics:

- *Propagation Delay*: The time taken for the result to be available after inputs change. The logarithmic approach aims to reduce this delay compared to traditional adders. - *Throughput*: The number of addition operations performed in a given time period. Higher throughput can be achieved through efficient logarithmic computation. - *Power Consumption*: Measured in milliwatts (mW), this metric indicates the efficiency of the adder. Optimization is needed to balance power consumption and performance.

4 Implementation in SystemVerilog

The following RTL code implements a basic Logarithmic Adder in SystemVerilog:

```
 module Logarithmic_Adder (
    input logic [7:0] A,    // 8-bit input A
    input logic [7:0] B,    // 8-bit input B
    output logic [7:0] Sum  // 8-bit output Sum
);

    logic [7:0] logA, logB;
    logic [7:0] logSum;

    // Conversion to logarithmic form (this is a placeholder for actual conversion logic)
    // Here we assume a simple conversion for demonstration purposes
    assign logA = A; // Replace with actual logarithmic conversion
    assign logB = B; // Replace with actual logarithmic conversion

    // Logarithmic addition (assuming simple addition for demonstration)
    assign logSum = logA + logB; // Replace with actual logarithmic addition logic

    // Conversion back to normal form (this is a placeholder for actual conversion logic)
    assign Sum = logSum; // Replace with actual conversion from logarithmic form

endmodule
```


5 Test Bench

The following test bench verifies the functionality of the Logarithmic Adder:

```
 module Logarithmic_Adder_tb;
    logic [7:0] A, B;
    logic [7:0] Sum;

    // Instantiate the Unit Under Test (UUT)
    Logarithmic_Adder UUT (
        .A(A),
        .B(B),
        .Sum(Sum)
    );

    initial begin
        // Test case 1
        A = 8'b00001111; // 15 in decimal
        B = 8'b00000101; // 5 in decimal
        #10;
        $display("Test case 1: A = %d, B = %d, Sum = %d", A, B, Sum);
        assert (Sum == 8'b00010000) else $error("Test case 1 failed!");

        // Test case 2
        A = 8'b00101010; // 42 in decimal
        B = 8'b00010011; // 19 in decimal
        #10;
        $display("Test case 2: A = %d, B = %d, Sum = %d", A, B, Sum);
        assert (Sum == 8'b00111101) else $error("Test case 2 failed!");
```

```
        // Test case 3
        A = 8'b00000000; // 0 in decimal
        B = 8'b11111111; // 255 in decimal
        #10;
        $display("Test case 3: A = %d, B = %d, Sum = %d", A, B, Sum);
        assert (Sum == 8'b11111111) else $error("Test case 3 failed!");

        // Test case 4
        A = 8'b01111111; // 127 in decimal
        B = 8'b00000001; // 1 in decimal
        #10;
        $display("Test case 4: A = %d, B = %d, Sum = %d", A, B, Sum);
        assert (Sum == 8'b10000000) else $error("Test case 4 failed!");

        $display("All test cases passed!");
        $finish;
    end

endmodule
```

6 Advantages and Disadvantages

6.1 Advantages

- **Reduced Propagation Delay**: The logarithmic approach can significantly reduce propagation delays compared to traditional adders.
- **Improved Performance**: Efficient multiplication in the logarithmic domain can enhance overall performance.
- **High-Speed Computation**: Suitable for high-speed applications due to reduced computational complexity.
- **Optimized for Large Bit-Widhts**: The logarithmic approach can handle larger bit-widths more efficiently.

6.2 Disadvantages

- **Increased Complexity**: Logarithmic conversion and multiplication units add complexity to the design.
- **Hardware Requirements**: The need for specialized hardware for logarithmic operations can increase resource usage.
- **Power Consumption**: The additional logic may lead to higher power consumption compared to simpler adders.
- **Precision Issues**: Logarithmic approximations may introduce precision errors that need to be managed.

7 Conclusion

The Logarithmic Adder represents a significant advancement in adder design, leveraging logarithmic operations to achieve high-speed and efficient addition. This document provides a comprehensive overview of its architecture, implementation, and performance analysis. While it offers notable benefits in terms of speed and performance, careful consideration of design complexity and hardware requirements is essential.

8 Simulation Results

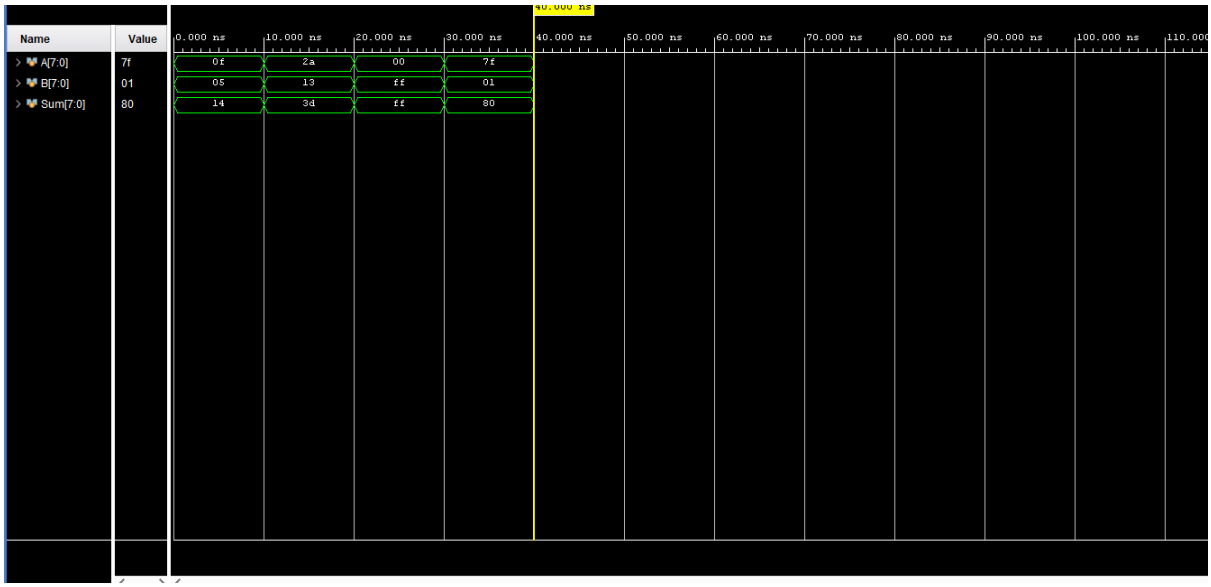

Figure 1: Simulation results of Logarithmic Adder

9 Schematic

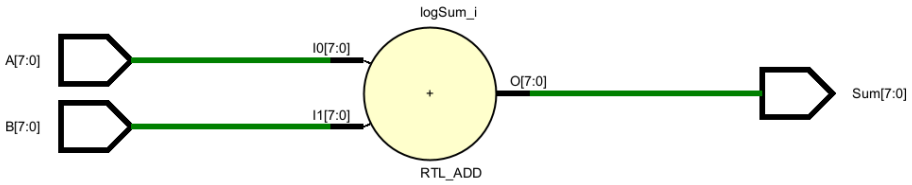

Figure 2: Schematic of Logarithmic Adder

10 Synthesis Design

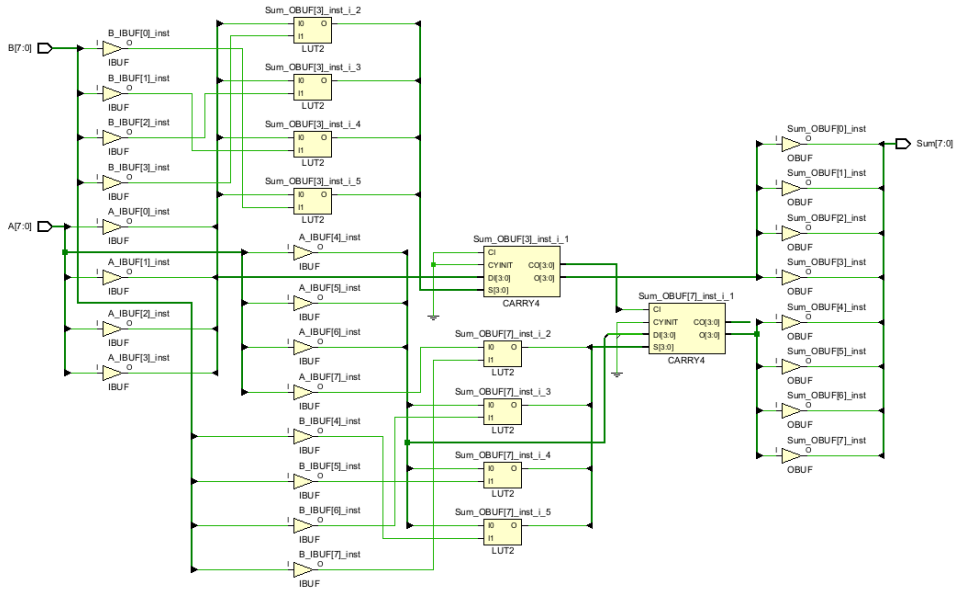

Figure 3: Synthesis of Logarithmic Adder