Project 3: Carry Lookahead Adder

A Comprehensive Study of Advanced Digital Circuits

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

Contents

1 Project Overview

In this project, we design and implement a Carry Lookahead Adder (CLA) using SystemVerilog. The CLA is known for its speed and efficiency in performing binary addition by overcoming the delay issues found in ripple carry adders.

2 Carry Lookahead Adder

2.1 Description

A Carry Lookahead Adder improves the speed of binary addition by calculating carry signals in advance, based on the input signals. Unlike ripple carry adders, which generate each carry sequentially, the CLA uses generate (G) and propagate (P) signals to determine carries in parallel, significantly reducing the propagation delay.

3 Why Choose a Carry Lookahead Adder?

The CLA is chosen due to its ability to perform fast arithmetic operations, making it ideal for high-speed computing applications. It reduces the overall delay compared to other types of adders, such as the ripple carry adder, making it a preferred choice in performance-critical digital systems.

3.1 RTL Code

Listing 1: 4 bit Carry Lookahead Adder

```
module CLA_4bit (
    input  logic [3:0] A,
    input  logic [3:0] B,
    input  logic       Cin,
    output logic [3:0] Sum,
    output logic       Cout
);

    logic [3:0] P, G, C;

    // Generate Propagate and Generate signals
    assign P = A ^ B;          // Propagate
    assign G = A & B;          // Generate

    // Carry Lookahead logic
    assign C[0] = Cin;
    assign C[1] = G[0]  (P[0] & C[0]);
    assign C[2] = G[1]  (P[1] & G[0])  (P[1] & P[0] & C[0]);
    assign C[3] = G[2]  (P[2] & G[1])  (P[2] & P[1] & G[0])  (P[2] &
        P[1] & P[0] & C[0]);

    // Sum and Carry-out
    assign Sum = P ^ C[3:0];
    assign Cout = G[3]  (P[3] & C[3]);

endmodule
```

3.2 Testbench

Listing 2: 4 bit Carry Lookahead Adder Testbench

```
module tb_CLA_4bit;

    // Declare inputs as reg and outputs as wire
    reg  [3:0] A;
    reg  [3:0] B;
    reg  Cin;
    wire [3:0] Sum;
    wire Cout;

    // Instantiate the CLA_4bit module
    CLA_4bit uut (
        .A(A),
        .B(B),
        .Cin(Cin),
        .Sum(Sum),
        .Cout(Cout)
    );

    // Testbench procedure
    initial begin
        // Test Case 1: Add zero to zero
        A = 4'b0000;
        B = 4'b0000;
        Cin = 1'b0;
        #10;
        $display("TC1: A = %b, B = %b, Cin = %b | Sum = %b, Cout =
            %b", A, B, Cin, Sum, Cout);

        // Test Case 2: Add two small numbers
        A = 4'b0011;  // 3
        B = 4'b0101;  // 5
        Cin = 1'b0;
        #10;
        $display("TC2: A = %b, B = %b, Cin = %b | Sum = %b, Cout =
            %b", A, B, Cin, Sum, Cout);

        // Test Case 3: Add two numbers with carry in
        A = 4'b1101;  // 13
        B = 4'b0111;  // 7
        Cin = 1'b1;
        #10;
        $display("TC3: A = %b, B = %b, Cin = %b | Sum = %b, Cout =
            %b", A, B, Cin, Sum, Cout);

        // Test Case 4: Add random values
        A = 4'b1010;  // 10
        B = 4'b0110;  // 6
        Cin = 1'b0;
        #10;
        $display("TC4: A = %b, B = %b, Cin = %b | Sum = %b, Cout =
            %b", A, B, Cin, Sum, Cout);

        // Test Case 5: Overflow case
        A = 4'b1111;  // 15
        B = 4'b1111;  // 15
        Cin = 1'b0;
        #10;
```

```
54          $display("TC5: A = %b, B = %b, Cin = %b | Sum = %b, Cout =
                %b", A, B, Cin, Sum, Cout);
55
56          // Finish simulation
57          $finish;
58      end
59
60  endmodule
```

4 How it works ?

The CLA works by using generate (G) and propagate (P) signals to calculate carry outputs in parallel. The generate signal is high if both corresponding bits of the inputs are high, ensuring a carry is generated. The propagate signal is high if at least one of the corresponding bits is high, ensuring the carry is propagated. This parallel calculation minimizes the delay compared to sequential carry generation in ripple carry adders.

4.1 Carry Generate (G) and Propagate (P) Table

A	B	Generate (G)	Propagate (P)	Carry (Cout)
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1

Table 1: Carry Generate (G) and Propagate (P) Values for Each Bit Pair

Explanation

In digital circuits, the terms "generate" and "propagate" are used to describe how the carry output (Cout) is calculated in adders.

Generate (G): The generate signal is high (1) when both bits A and B are high. This means that a carry will be generated regardless of the input carry (Cin).

$$G = A \cdot B$$

Propagate (P): The propagate signal is high (1) when at least one of the bits A or B is high. This means that the carry input (Cin) will propagate through to the carry output (Cout).

$$P = A + B$$

Carry Output (Cout): The carry output is determined using both the generate and propagate signals. If either G is high or both P and the input carry (Cin) are high, the carry output will be high.

$$Cout = G + (P \cdot Cin)$$

Table Interpretation: - When A and B are both 0, neither generate nor propagate signals are activated, so Cout is 0. - When A is 0 and B is 1 (or vice versa), the propagate signal is activated but not the generate signal, so Cout depends on Cin. - When both A and B are 1, both generate and propagate signals are activated, resulting in Cout being 1 regardless of Cin.

This table and explanation help in understanding how the generate and propagate signals are used to calculate carry outputs in adders, improving the speed and efficiency of binary addition.

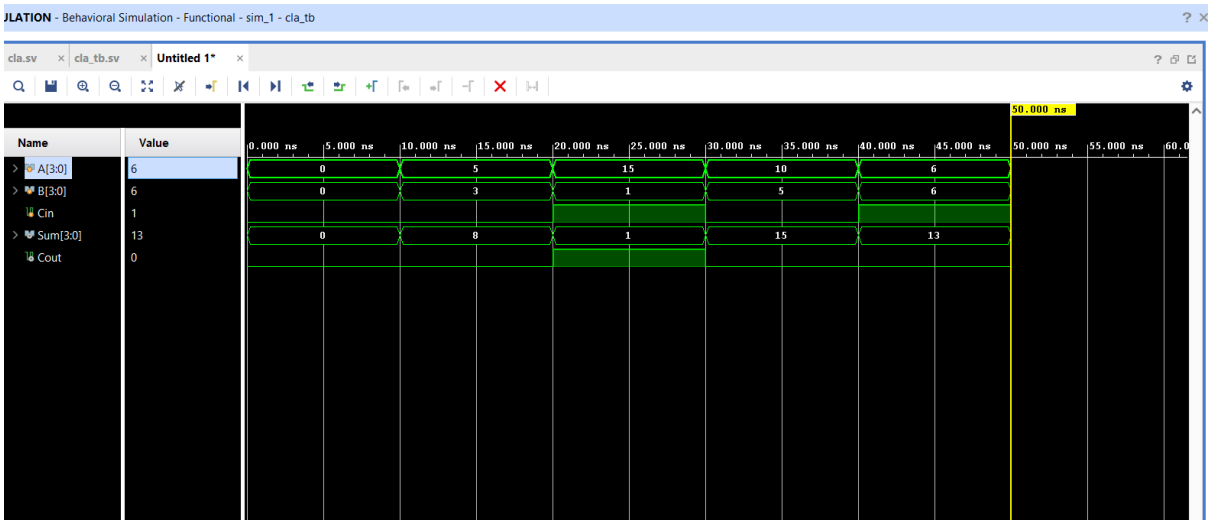

Figure 1: Simulation results of 4 Bit CLA

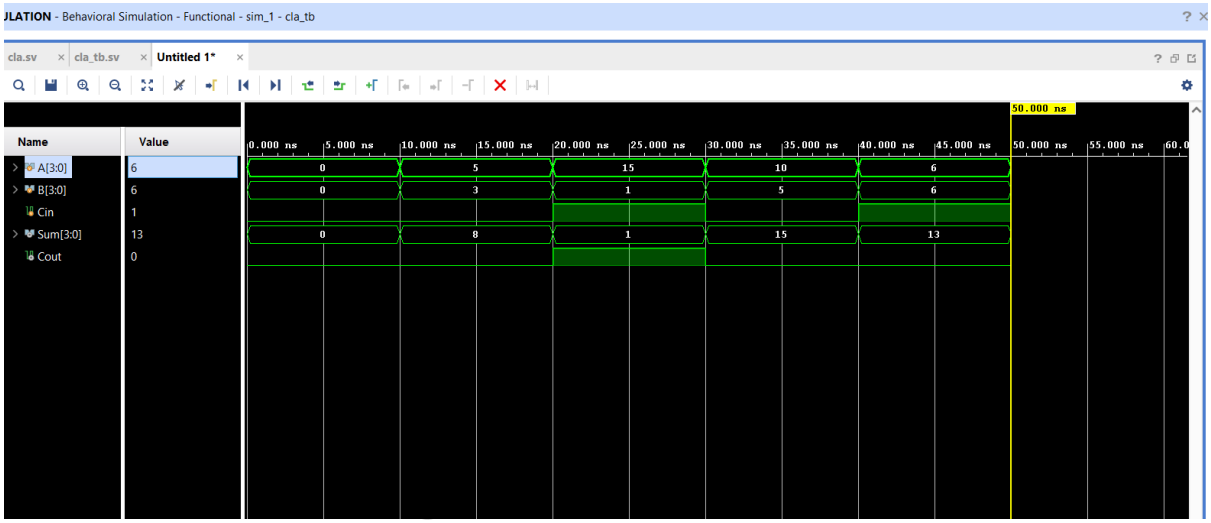

Figure 2: Synthesis Design of 4 Bit CLA

4.2 Simulation Results

4.3 Synthesis Design

4.4 Schematic

4.5 Advantages

- Faster computation compared to ripple carry adders.
- Reduced propagation delay.
- Efficient for large bit-width additions.

4.6 Disadvantages

- More complex circuitry.
- Increased area and power consumption due to additional logic.

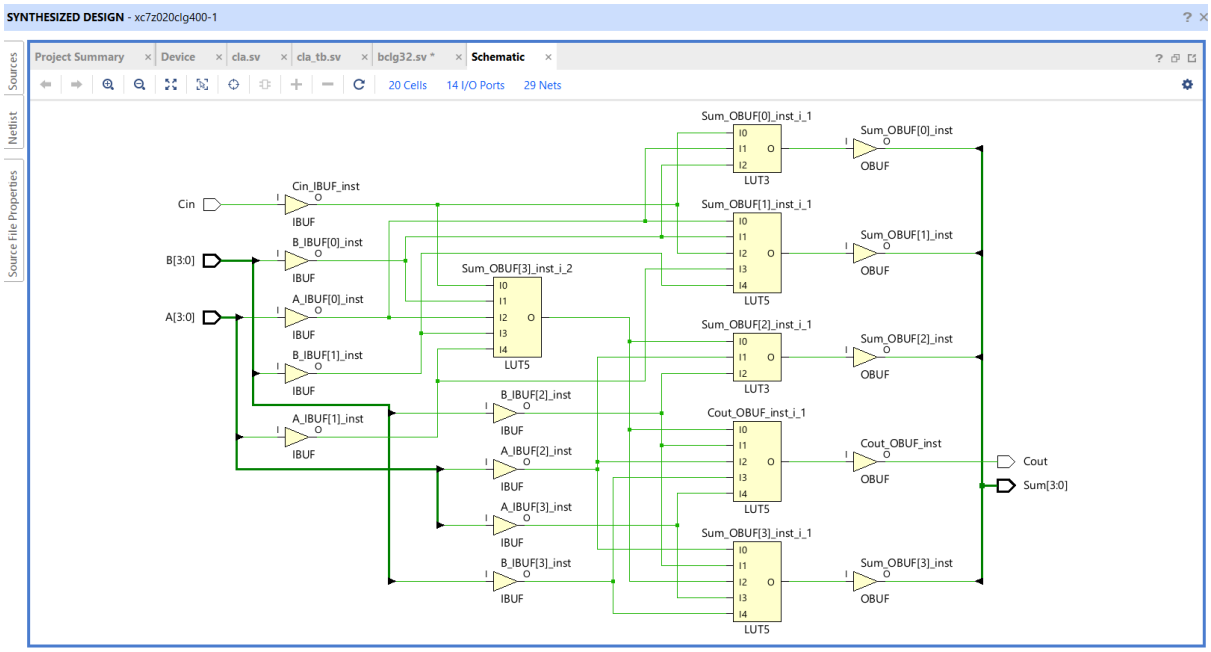

Figure 3: Schematic of 4 Bit CLA

4.7 Applications

- High-speed arithmetic circuits.
- Microprocessors.
- Digital signal processing (DSP) systems.
- Computer graphics.