Hardware Lab (CS 224)
Assignment 3

Group No. 18

Team Members:
Daksh Agarwal (230101123)
Aditya Shukla (230101115)
Kartik Maheshwari (230101116)
Ryan Aggarwal (230101117)

February 11, 2025

Contents

1 Objective

The objective of this lab is to implement and analyze different types of adders, including Carry Look Ahead Adder (CLA) and Carry Save Adder (CSA), in Verilog and on hardware platforms like FPGA and Breadboard.

2 Question 1: 4-bit Carry Look Ahead Adder in Verilog

2.1 Implementation

A modular Verilog implementation of a 4-bit two-input Carry Look Ahead Adder (CLA) is provided in the zip file along with the report named as **CLA_4bit.v**. Some implementation details are given as follows :

- The code is made as modular as possible.
- CLA_4bit module:This module efficiently calculates the 4-bit sum and carry-out using look-ahead logic to determine the carry of each bit position in advance, reducing the delay typical in sequential carry calculations. This makes it significantly faster than simpler sequential carry adders for applications where speed is critical, such as in high-performance computing systems.

2.2 Test Bench and Simulation

The test bench verifies the correctness of the CLA design using various input cases. The test bench file is again provided along with the report in the zip file named as **CLA_4bit_tb.v**.

- The testbench first instantiates the CLA_4bit module.
- The time unit is set to 1 ns with a precision of 1 ps.
- The inputs A, B, and Cin are changed sequentially with a delay of 10 ns between each set of assignments.

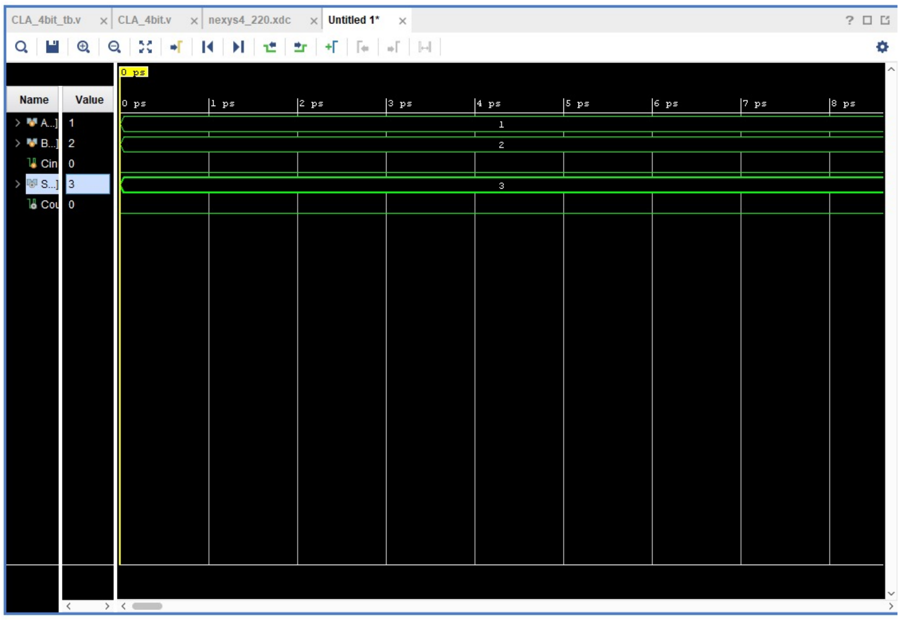

Figure 1: Simulation

2.3 Synthesis and FPGA Demonstration

The design is synthesized and simulated using Vivado/Xilinx ISE. The synthesized bit file is downloaded to the FPGA for demonstration.

3 Question 2: 3-bit Carry Look Ahead Adder on Breadboard

3.1 Implementation

A 3-bit two-input Carry Look Ahead Adder (CLA) is implemented on a breadboard using logic gates. This adder improves speed over the traditional ripple carry adder by generating carry bits in advance using combinational logic.

3.2 Explanation of Carry Look Ahead Adder

A Carry Look Ahead Adder (CLA) is designed to reduce the delay caused by carry propagation in conventional adders. Instead of waiting for the carry to propagate through all bits, the CLA adder computes the carry values using the generate (G) and propagate (P) terms. This allows the sum to be computed in parallel, significantly improving speed.

By using this approach, the delay is minimized to a few gate delays rather than increasing linearly with the number of bits, making CLA a preferred choice for high-speed arithmetic operations.

3.3 Carry Look Ahead Expressions

The CLA adder uses the following equations to determine the sum and carry:

Carry Generate and Propagate Terms:

$$G_i = A_i \cdot B_i \tag{1}$$

$$P_i = A_i \oplus B_i \tag{2}$$

Carry Outputs:

$$C_1 = G_0 + (P_0 \cdot C_0) \tag{3}$$

$$C_2 = G_1 + (P_1 \cdot G_0) + (P_1 \cdot P_0 \cdot C_0) \tag{4}$$

$$C_3 = G_2 + (P_2 \cdot G_1) + (P_2 \cdot P_1 \cdot G_0) + (P_2 \cdot P_1 \cdot P_0 \cdot C_0) \tag{5}$$

Sum Outputs:

$$S_i = P_i \oplus C_i \tag{6}$$

3.4 Gates Used

The logic gates used in the circuit include XOR, AND, and OR gates, which are essential for generating the propagate and generate signals.

3.5 ICs Used

The following ICs are used for implementation:

- 4 input AND IC - 7421 (1)
- 3 input AND IC - 7415 (1)
- 2 input XOR IC - 7486 (2)
- 2 input OR IC - 7432 (2)
- 2 input AND IC - 7408 (2)

3.6 Circuit Diagram

The following circuit diagram illustrates the implementation of the 3-bit CLA adder:

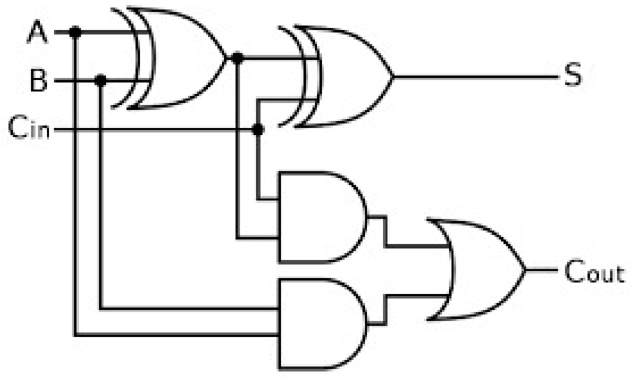

Figure 2: Full Adder Circuit Using Basic Logic Gates

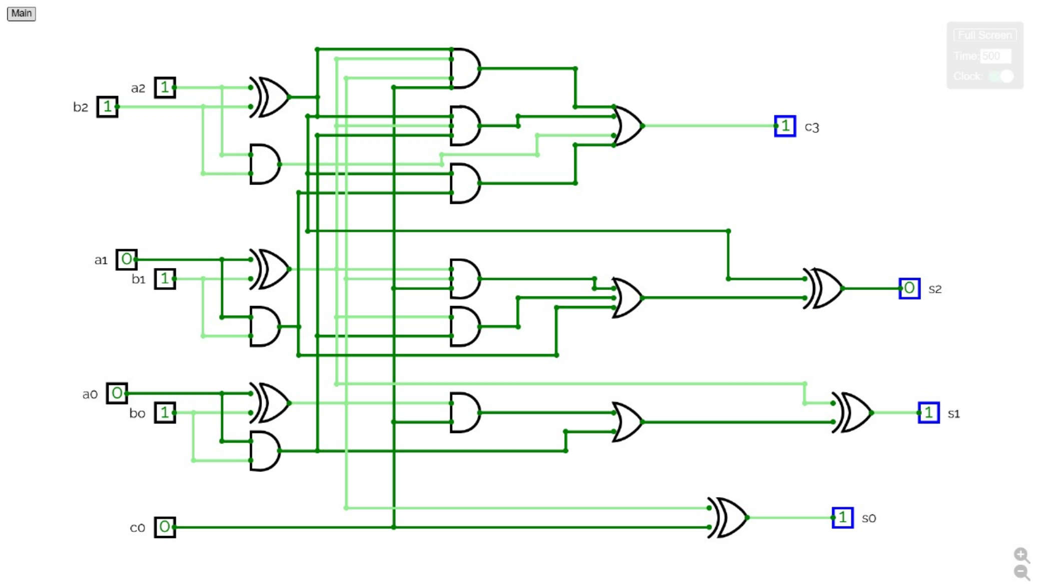

Figure 3: 3-bit Carry Look Ahead Adder Circuit

3.7 Circuit Photo

Following is the Circuit Photo for the implementation in which all live connections can be seen.

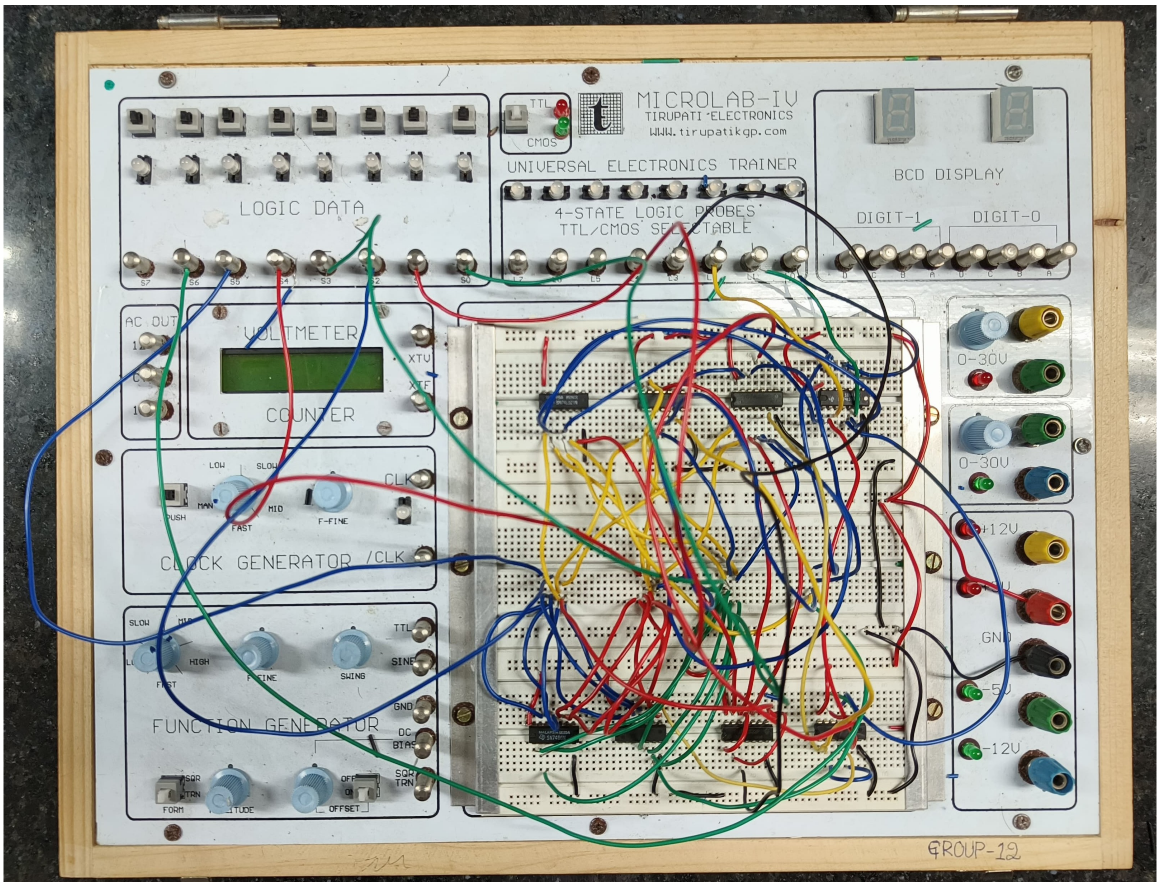

Figure 4: Full Adder Circuit Using Basic Logic Gates

4 Question 3: 32-bit Carry Save Adder in Verilog

4.1 Implementation

A modular Verilog implementation of a 32-bit three-input Carry Save Adder (CSA) is provided in the zip file along with the report named as **name.v**. Some implementation details are given as follows :

- The code is made as modular as possible.
- fulladder Verilog module calculates both the sum and the carry of two binary numbers and a carry-in bit, which can be used as part of a larger binary addition operation by cascading multiple such full adder modules. This is fundamental in designing arithmetic circuits that handle larger binary numbers.
- A CSA_32bit module loops 32 times through the bits and adds them saving the carries.
- The RAC_32Bit module then does the ripple carry adder on the carries and the resulting sum.

- Finally the CSA_32Bit_with_RCA module calls all the above stated modules.

4.2 Test Bench and Simulation

The test bench verifies the correctness of the CLA design using various input cases. The test bench file is again provided along with the report in the zip file named as **Hardware_Lab_2_tb.v**.

- The testbench first instantiates the CSA_32Bit_with_RCA module module.
- The time unit is set to 1 ns.
- The bits for addition are changed in a cycle of 10 ns.

The simulation details are given below. The cycle is 10 ns along with Cin, A, and B bits changing can be observed.

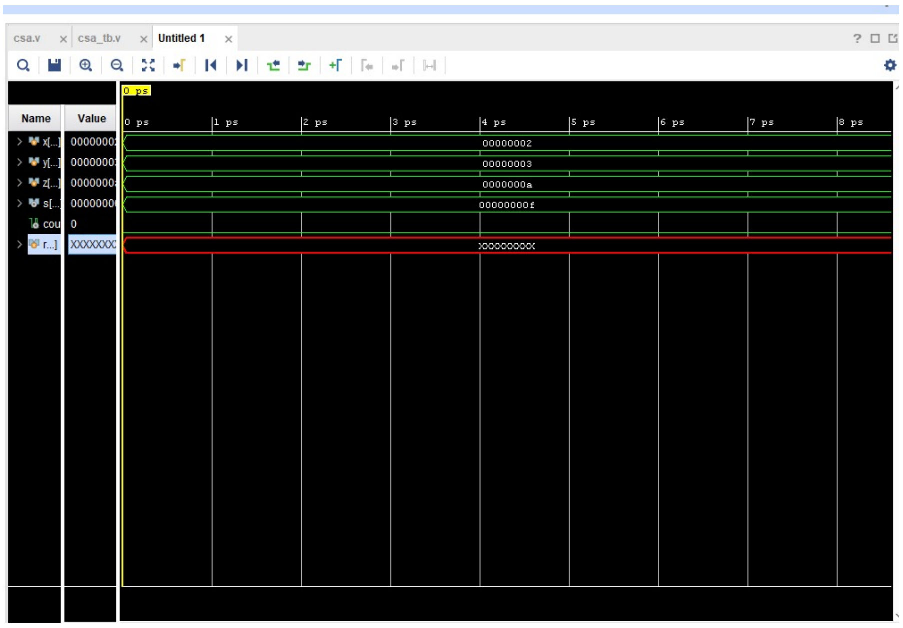

Figure 5: Simulation

5 Comparison of 32-bit 3-Input Carry-Save Adder and Ripple Carry Adder

In this section, we compare the **area** and **delay** of a 32-bit three-input *Carry-Save Adder (CSA)* with a 32-bit three-input *Ripple Carry Adder (RCA)*.

5.1 Delay Analysis

The delay of an adder is primarily determined by how carries propagate through the circuit.

5.1.1 Ripple Carry Adder (RCA) Delay Analysis

A **Ripple Carry Adder (RCA)** performs addition by propagating the carry sequentially through all the bit positions. In the case of a **32-bit 3-input RCA**, the addition is performed in two stages:

1. The first two inputs are summed using a 32-bit RCA.

2. The sum from the previous step is added to the third input using another 32-bit RCA.

The worst-case delay is given by:

$$T_{RCA} = 2 \times (32 \times T_{FA}) \tag{7}$$

where T_{FA} is the delay of a full adder.

5.1.2 Carry-Save Adder (CSA) Delay Analysis

A **Carry-Save Adder (CSA)** computes sums in parallel without propagating carry immediately. The structure consists of:

1. A layer of 32 full adders that computes sum and carry separately.

2. A 32-bit **Ripple Carry Adder (RCA)** to process the final sum.

The worst-case delay is:

$$T_{CSA} = T_{FA} + (32 \times T_{FA}) \tag{8}$$

where the first term T_{FA} corresponds to the CSA operation, and the second term corresponds to the final RCA addition.

5.1.3 Delay Comparison

Comparing the delays:

$$T_{RCA} = 64 \times T_{FA} \tag{9}$$

$$T_{CSA} = 33 \times T_{FA} \tag{10}$$

Since $33T_{FA}$ is significantly smaller than $64T_{FA}$, we conclude that the Carry-Save Adder is **almost twice as fast** as the Ripple Carry Adder.

5.2 Area Analysis

The area of an adder is determined by the number of full adders (FAs) used.

5.2.1 Ripple Carry Adder (RCA) Area Analysis

A 32-bit RCA requires **32 full adders** per addition. Since a three-input RCA requires two sequential adders, the total number of full adders used is:

$$A_{RCA} = 2 \times 32 = 64 \text{ full adders.} \tag{11}$$

5.2.2 Carry-Save Adder (CSA) Area Analysis

A 3-input CSA consists of a single layer of 32 full adders for sum and carry computation, followed by a 32-bit RCA to finalize the result. The total area requirement is:

$$A_{CSA} = 32 + 32 = 64 \text{ full adders.} \tag{12}$$

5.2.3 Area Comparison

Since both adders require the same number of full adders, their area usage is identical:

$$A_{RCA} = A_{CSA} = 64 \text{ full adders.} \tag{13}$$

5.3 Summary of Comparison

Table 1 summarizes the differences between the two adders.

Metric	**3-input RCA (32-bit)**	**3-input CSA (32-bit)**	**Comparison**
Delay	$64 \times T_{FA}$	$33 \times T_{FA}$	CSA is $\approx 2\times$ faster
Area	64 Full Adders	64 Full Adders	Same Area

Table 1: Comparison of 32-bit 3-input Ripple Carry Adder and Carry-Save Adder

5.4 Conclusion

From the comparison, we observe that the **Carry-Save Adder (CSA) is significantly faster** than the Ripple Carry Adder (RCA) due to its reduced carry propagation. However, both adders require the **same area** in terms of the number of full adders. This makes the CSA a more efficient choice for high-speed arithmetic operations.

6 Question 4: Gate Count for 32-bit Three-input CLA

A Carry Look Ahead Adder (CLA) works by reducing the carry propagation delay using generate (G) and propagate (P) functions. The number of logic gates required for a 32-bit three-input CLA can be derived as follows:

6.1 Calculation and Analysis

For the ith bit, the propagate signal p_i and generate signal g_i generated by it and used by the (i+1)th bit is (each requiring 1 two input gate)

$$p_i = a_i \oplus b_i$$

$$g_i = a_i b_i$$

For each carry signal c_i to be used to calculate the (i+1)th sum, we have

$$c_i = g_{-1}p_0p_1 \cdots p_{i-1} + g_0p_1p_2 \cdots p_{i-1} + g_1p_2p_3 \cdots p_{i-1} + \cdots + g_{i-2}p_{i-1} + g_{i-1}$$

Note that $g_{-1} = c_0$

For each product term of length n, n-1 two input AND gates are required. And to sum n product terms, n-1 two input OR gates are required. Therefore, we have

$$\underbrace{i}_{\text{OR Gate}} + \underbrace{\frac{i(i+1)}{2}}_{\text{AND Gates}}$$

$$\sum_{i=1}^{n} \frac{i(i+3)}{2} = \frac{1}{6}n(n+1)(n+5)$$

To calculate sum s_i, p_i and c_i and 1 two input XOR gate is required.

So total, we have

$$Gates = 3n + \frac{1}{6}n(n+1)(n+5)$$

Substituting n = 32, the number of two input gates required is 6608 * 2 = 13216. (Because of the third input in the CLA)

7 Conclusion

This lab provided extensive hands-on experience in implementing and analyzing various adder architectures. The 4-bit Carry Look Ahead Adder (CLA) and the 32-bit Carry Save Adder (CSA) were implemented using Verilog and synthesized using the Vivado/Xilinx ISE tool. The designs were simulated to verify their correctness and then synthesized into FPGA hardware for real-world demonstration. Additionally, a 3-bit CLA was constructed on a breadboard using logic gates to understand its practical implementation.

Through this lab, we explored the efficiency of different adder architectures by comparing their area and delay. The gate count analysis for a 32-bit three-input CLA was also done. The comparison with other adder architectures, such as Ripple Carry Adders, highlighted the advantages of using CLA in terms of speed improvement.