Design and Optimization of a 4-Bit Carry Look-Ahead Adder Using NAND and XOR Gates

Gautam Gandhi
Department of Electronics and Communication Engineering
International Institute of Information Technology, Hyderabad
Hyderabad, Telangana, India
Email: gautam.gandhi@students.iiit.ac.in

***Abstract*—** This report presents a 4-bit carry look-ahead (CLA) adder optimized using NAND and XOR gates to reduce transistor count, minimize propagation delay, and improve overall circuit efficiency. Based on the fastest CLA design methodology by Pai and Chen, the implementation leverages a recursive structure for carry calculation, which minimizes the cascading delay common in adder designs. Simulations were conducted in NGSPICE using a 180nm technology node, and layout was performed in MAGIC. Results from post-layout extractions confirm the expected performance improvements, particularly in terms of delay and transistor efficiency.

***Keywords*—** CLA adder, high-speed adder, NGSPICE, transistor optimization, CMOS, NAND gate, XOR gate, integrated circuit, gate delay.

I. INTRODUCTION

Adders are fundamental building blocks in digital systems, playing a critical role in arithmetic operations within processors. The carry look-ahead (CLA) adder offers a high-speed solution by reducing the delay caused by carry propagation, an issue seen in simpler designs like the ripple-carry adder. This project focuses on implementing a 4-bit CLA adder optimized for speed and transistor efficiency, following the improved CLA methodology presented by Pai and Chen in "The Fastest Carry Lookahead Adder" [1].

The primary objective was to design a CLA adder using NAND and XOR gates only, thereby reducing the number of transistors compared to traditional CLA implementations, which use a mix of logic gates. This implementation is tested with NGSPICE and laid out in MAGIC, followed by post-layout simulations for performance verification.

II. CARRY LOOK-AHEAD ADDER DESIGN AND OPTIMIZATION

In digital arithmetic circuits, especially those used for high-speed applications, the delay associated with carry propagation is a critical bottleneck. The **Carry Look-Ahead Adder (CLA)** is designed to overcome this delay by generating the carry signals in parallel, thus significantly reducing the time required to perform addition. In this section, we discuss the basic implementation of the CLA and present an optimized design approach using only **NAND** and **XOR** gates to reduce transistor count and improve performance.

2.1 Standard Implementation of the Carry Look-Ahead Adder

The CLA adder relies on two fundamental concepts:

- **Generate (g_i):** This signal indicates that a particular bit pair (a_i, b_i) will generate a carry, regardless of any incoming carry from the previous bit. It is defined as:
$$g_i = a_i \cdot b_i$$
- **Propagate (p_i):** This signal indicates that a carry-in will be propagated through the bit pair (a_i, b_i) to the next bit position. It is defined as:
$$p_i = a_i \oplus b_i$$

With these signals, we can recursively calculate each carry output C_{i+1} in terms of the generate and propagate signals and the initial carry input C_0. The recursive formula for each carry bit is given by:

$$C_{i+1} = g_i + (p_i \cdot C_i)$$

where $+$ represents the OR operation, and $\cdot$ represents the AND operation.

Using this formula, the carry bits for a 4-bit CLA can be expanded as:

- $C_1 = g_0 + (p_0 \cdot C_0)$
- $C_2 = g_1 + (p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot C_0)$
- $C_3 = g_2 + (p_2 \cdot g_1) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot p_1 \cdot p_0 \cdot C_0)$
- $C_4 = g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) + (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0)$

This approach reduces the propagation delay by calculating each carry bit in parallel, but it still requires multiple gates for each carry signal, increasing the circuit complexity and the number of transistors.

2.2 Optimized Design Using NAND and XOR Gates

To further optimize the design, we can reduce the number of transistors by implementing the CLA adder using only NAND and XOR gates, which are transistor-efficient in CMOS technology. This method leverages the fact that NAND gates are faster and require fewer transistors compared to AND, OR, and NOR gates. In CMOS logic, each 2-input NAND gate typically requires 4 transistors, while XOR gates, though slightly more complex, provide an efficient way to implement the propagate function.

Derivation of Carry Expressions Using NAND Gates:

The carry-out expression for the CLA can be restructured to use NAND gates by applying De Morgan's laws. We begin with the recursive expression for C_{i+1} :

$$C_{i+1} = g_i + (p_i \cdot C_i)$$

Applying De Morgan's theorem, this expression can be transformed to:

$$C_{i+1} = \overline{\overline{g_i} \cdot \overline{p_i \cdot C_i}}$$

which can be implemented as:

$$C_{i+1} = \text{NAND}(g_i', \text{NAND}(p_i, C_i))$$

where g_i' is the inverted generate signal, i.e., $g_i' = \overline{g_i} = \text{NAND}(a_i, b_i)$.

This expression achieves the carry-out using just NAND gates per bit position, significantly reducing transistor count compared to traditional CLA implementations.

Using this NAND-based formulation, each carry bit can be computed in a recursive manner for a 4-bit CLA adder:

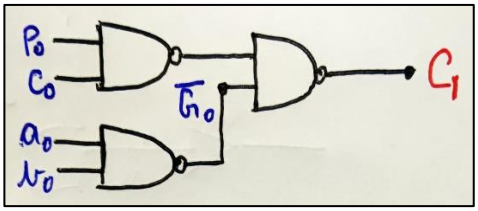
1. **First Carry C_1:**

$$C_1 = \overline{\overline{g_0} \cdot \overline{p_0 \cdot C_0}}$$

This expression can be rewritten using NAND gates as:

$$C_1 = \text{NAND}\left(g_0', \text{NAND}(p_0, C_0)\right)$$

where g_0' represents the NAND form of g_0, i.e., $g_0' = \overline{g_0} = \text{NAND}(a_0, b_0)$,
and $C_0 =$ input carry.

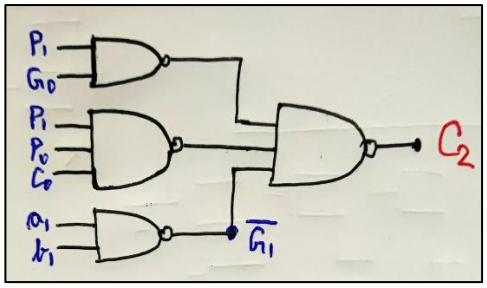


2. **Second Carry C_2:**

$$C_2 = \overline{\overline{g_1} \cdot \overline{p_1 \cdot g_0} \cdot \overline{p_1 \cdot p_0 \cdot C_0}}$$

Using NAND logic:

$$C_2 = \text{NAND}\left(g_1', \text{NAND}(p_1, g_0), \text{NAND}(p_1, p_0, C_0)\right)$$

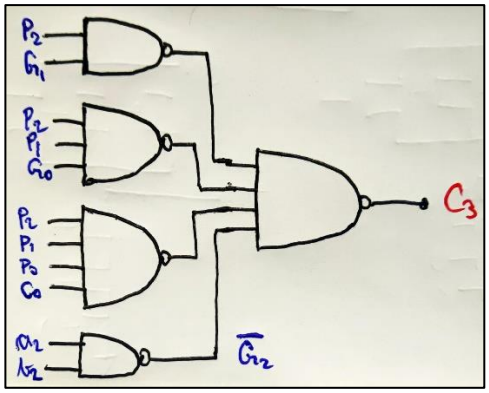


3. **Third Carry C_3:**

$$C_3 = \overline{\overline{g_2} \cdot \overline{p_2 \cdot g_1} \cdot \overline{p_2 \cdot p_1 \cdot g_0} \cdot \overline{p_2 \cdot p_1 \cdot p_0 \cdot C_0}}$$

Using NAND logic:

$$C_3 = \text{NAND}(g_2', \text{NAND}(p_2, g_1), \text{NAND}(p_2, p_1, g_0), \text{NAND}(p_2, p_1, p_0, C_0))$$

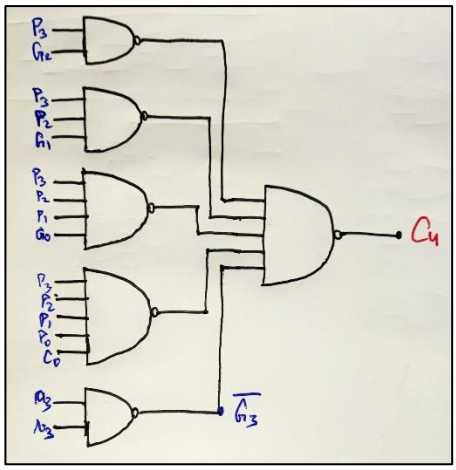


4. **Fourth Carry C_4:**

$$C_4 = \overline{\overline{g_3} \cdot \overline{p_3 \cdot g_2} \cdot \overline{p_3 \cdot p_2 \cdot g_1} \cdot \overline{p_3 \cdot p_2 \cdot p_1 \cdot g_0} \cdot \overline{p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0}}$$

This can be implemented using NAND gates as:

$$C_4 = \text{NAND}(g_3', \text{NAND}(p_3, g_2), \text{NAND}(p_3, p_2, g_1), \text{NAND}(p_3, p_2, p_1, g_0), \text{NAND}(p_3, p_2, p_1, p_0, C_0))$$



2.3 Sum Calculation Using XOR Gates

Once the carries are calculated, the sum for each bit position i can be computed using the propagate signal and the previous carry:

$$S_i = p_i \oplus C_i$$

To implement this efficiently, we use the optimized XOR gate design by Tripti Sharma et al. [14], which minimizes the number of transistors and offers faster switching times. This design computes the XOR function at each sum bit without requiring additional inverters or complex logic.

2.4 Summary of Optimization Benefits

By restructuring the carry logic to use only NAND and XOR gates, we achieve a more transistor-efficient design.

- **Reduced Transistor Count**: The NAND-based design minimizes the number of transistors required for each carry bit calculation.
- **Improved Speed**: NAND gates have lower propagation delays in CMOS, which directly contributes to a faster overall carry calculation.
- **Lower Power Consumption**: Fewer transistors lead to reduced capacitance and, thus, lower power consumption, making this design more energy efficient.

This design provides a high-speed solution with reduced complexity, making it ideal for high-performance arithmetic circuits in applications such as processors and digital signal processors (DSPs).

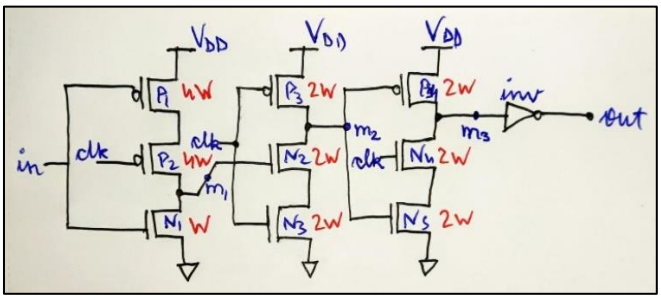
Overall, this optimized Carry Look-Ahead Adder design retains the parallel carry generation capability of the conventional CLA while reducing hardware cost, making it a valuable approach in integrated circuit (IC) design.

III. TOPOLOGY AND SIZING

In this section, we provide the detailed design, topology, and sizing of the components used in the Carry Look-Ahead Adder (CLA) circuit, focusing on both the adder modules and the D-flip-flop. These elements are critical in ensuring efficient performance in terms of speed, power consumption, and area.

3.1 D-Flip-Flop

The D-flip-flop is implemented using True Single-Phase Clocking (TSPC) technology. TSPC is an efficient design methodology that eliminates the need for complementary clock signals, thereby reducing power consumption, clock skew, and layout complexity. The TSPC-based flip-flop combines latches and inverters, ensuring high-speed operation with a reduced transistor count.



Sizing of PMOS and NMOS:

The transistors at each level are sized such that the equivalent width of the PUN is 2W and that of PDN is W.

Key Features of TSPC-Based Design:

- Fewer transistors are required compared to traditional master-slave flip-flops.
- The absence of complementary clock signals minimizes routing complexity.

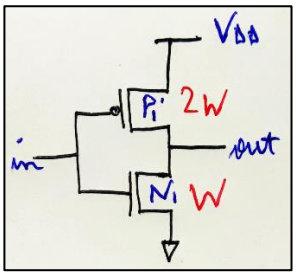
3.2 CLA Adder

The CLA adder topology is designed using the optimized logic for carry generation and propagation. The critical modules include NOT gates, XOR gates, and NAND gates.

Sizing of Gates:

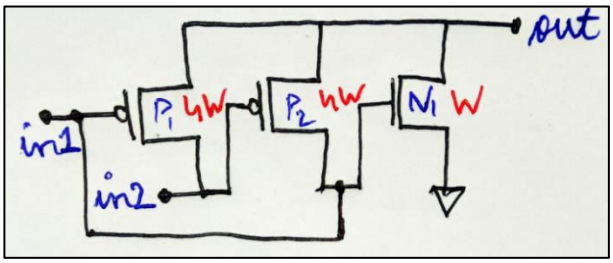
The sizing of all these gates is done such that the equivalent width of the PUN is 2W and that of PDN is W so as to ensure minimal delay and equal rise and fall times in the carry propagation path.

NOT Gate:



XOR Gate:

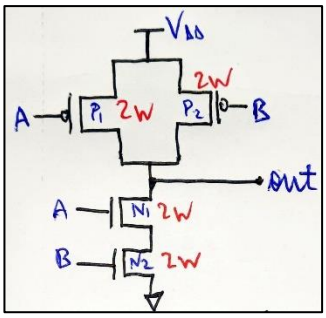
In this project, the XOR gate is implemented using a **3-transistor (3T) design**, which is based on the approach described in the research paper *"New Efficient Design for XOR Function on the Transistor Level"* by Tripti Sharma et al. [2]. This 3T XOR design is compact and optimized for low power consumption, reduced area, and improved speed, making it suitable for the high-performance requirements of the Carry Look-Ahead Adder.



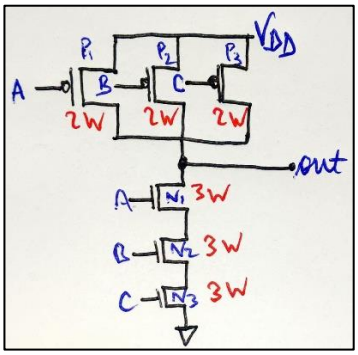
The circuit shown in the Fig. comprises of two PMOS and one NMOS. The output signal for inputs 01 and 11 will be complete. For AB= 10, both P2 and N1 will be on and will pass a poor "HIGH" signal level to the output, the reason is that when P2 and N1 will be on, the resistance of both devices will come in parallel and the total resistance will decrease which will lead to the degradation in output voltage and as the aspect ratio of P2 is larger than that of N1, the voltage at OUT terminal will be reflected as per the functioning of transistor P2. However, the threshold drop occurs across P2 for AB=00 and consequently the output Y degrades with respect to the input.

We put two inverters back-to-back before the final output to get perfect HIGH and LOW signals at the final output.

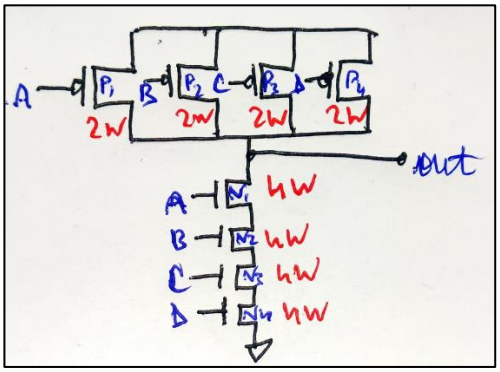
2-Input NAND Gate:



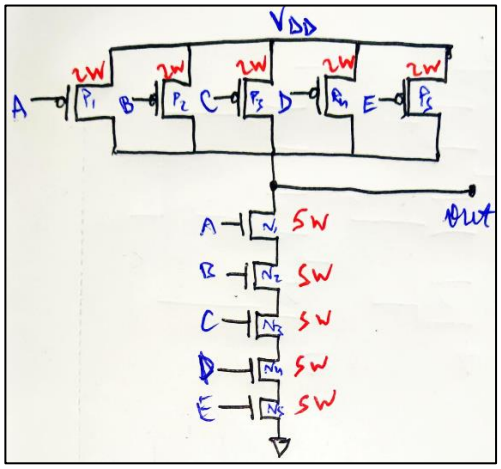
3-Input NAND Gate:



4-Input NAND Gate:



5-Input NAND Gate:

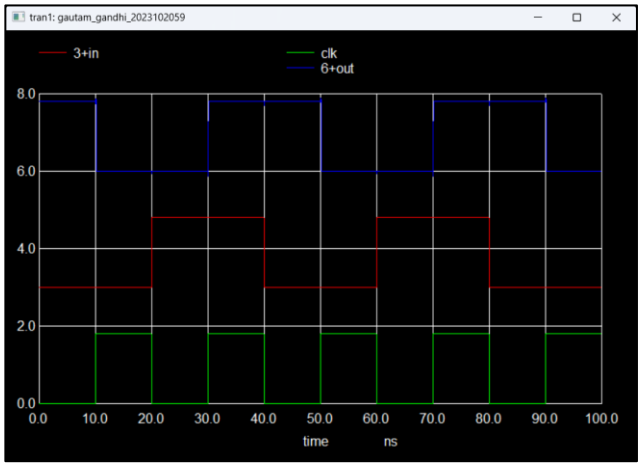


IV. NGSPICE SIMULATIONS

In this section, we will test the different blocks of our circuit using NGSPICE simulations.

4.1 D-Flip-Flop Testing

The flip-flop is tested by giving clock and input pulses having different time periods. The following plot confirms the correct functionality of our positive edge triggered D-Flip-Flop.

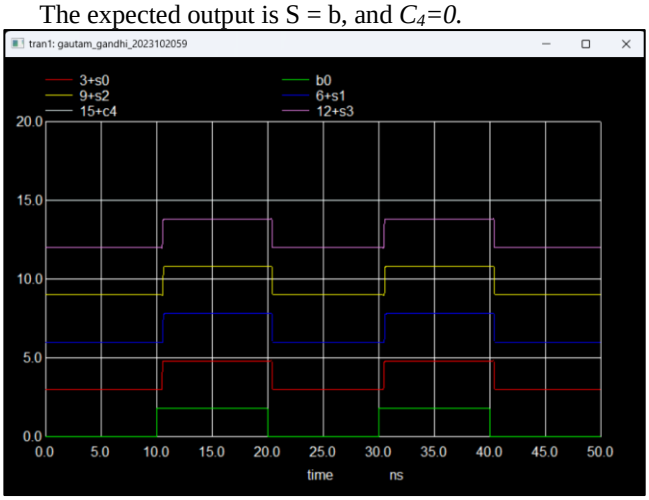


4.2 CLA Adder Testing

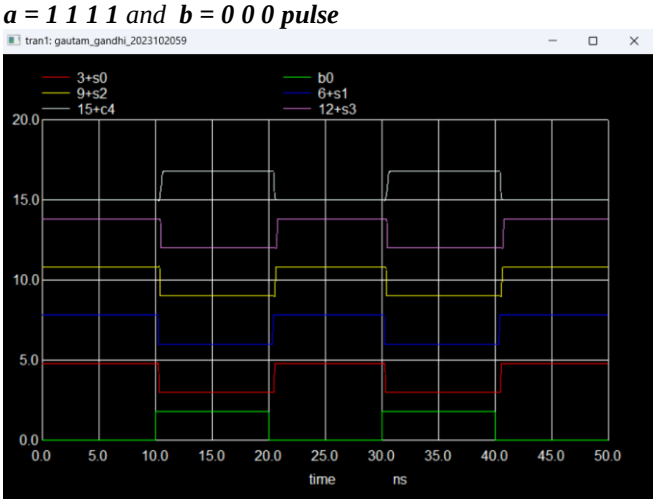
The CLA Adder is tested for different test cases:

- ***a = 0 0 0 0*** *and* ***b = pulse pulse pulse pulse***

The expected output is S = b, and C_4=*0.*



a = 1 1 1 1 *and* ***b = 0 0 0 pulse***



Conclusion:

The above plots confirm the correct functioning of the CLA Adder.

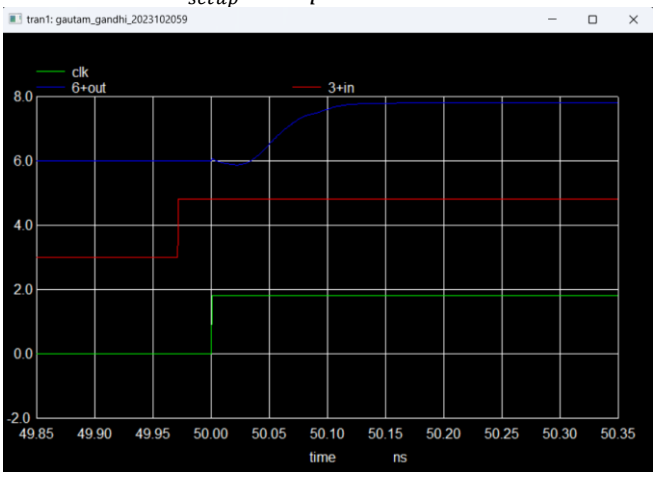
V. D-FLIP-FLOP PERFORMANCE ANALYSIS

The D-flip-flop is a critical component of the Carry Look-Ahead Adder (CLA) design, responsible for latching data during synchronous operations. This section details the extraction of key performance parameters—setup time, hold time, and clock-to-Q delay—from NGSPICE simulations.

The D-flip-flop circuit was simulated in NGSPICE with a supply voltage of $V_{DD} = 1.8\,V$, and a *2W/W* inverter was added as a load to model real-world conditions.

5.1 Setup Time (t_{setup}):

- It is the minimum time before the clock's rising edge that the D input must remain stable for the output (Q) to correctly latch the data.
- To calculate t_{setup}, the input D was varied closer to the clock edge until incorrect latching occurred.
- **Result:** $t_{setup} \approx 29\,ps$.



5.2 Hold Time (t_{hold}):

- It is the minimum time after the clock's rising edge that the D input must remain stable to avoid glitches or incorrect data at Q.
- To calculate t_{hold}, post-clock-edge variations in D were applied until latching failed.
- **Result:** $t_{hold} \approx 12\,ps$.

5.3 Clock-to-Q Delay (t_{clk-Q}):

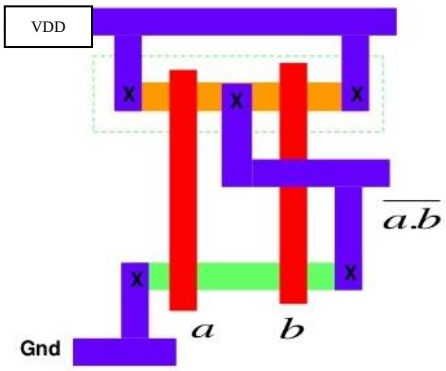
- It is the time it takes for the change in the clock signal (rising edge) to propagate to a change in the output Q.
- The delay was measured between the clock edge and the moment Q transitioned.
- **Result:** $t_{clk-Q} \approx 93\ ps$

Conclusion

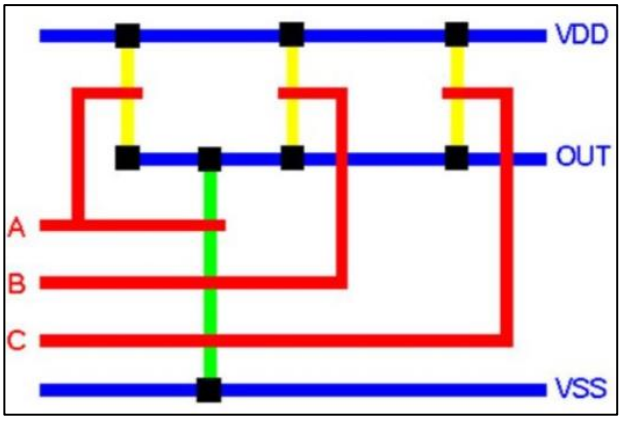
The NGSPICE simulation results confirm that the designed D-flip-flop meets the timing requirements for integration into the CLA adder. The **setup time (29 ps)**, **hold time (12 ps)**, and **clock-to-Q delay (93 ps)** demonstrate optimal performance for the given technology node. These metrics ensure reliable and high-speed operation in the overall design.

VI. STICK DIAGRAMS

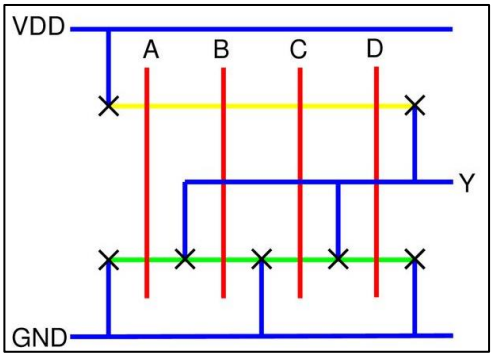
6.1 2-Input NAND



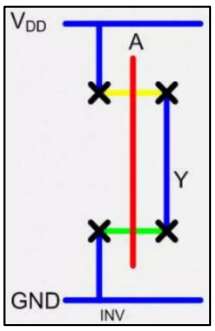
6.2 3-Input NAND



6.3 4-Input NAND



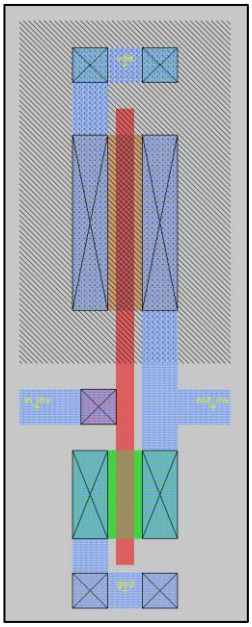
6.4 NOT Gate



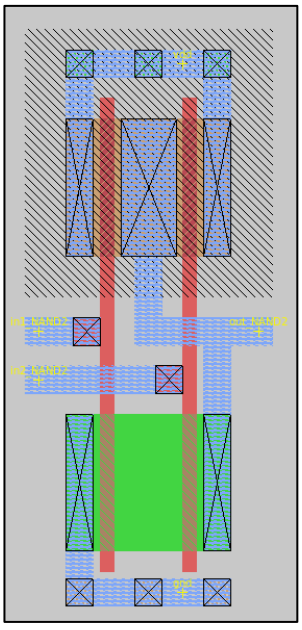
VII. LAYOUT DESIGN

This section details the physical layout design of the Carry Look-Ahead Adder (CLA) blocks, including D-flip-flops, XOR gates, and NAND gates, using the MAGIC layout editor.

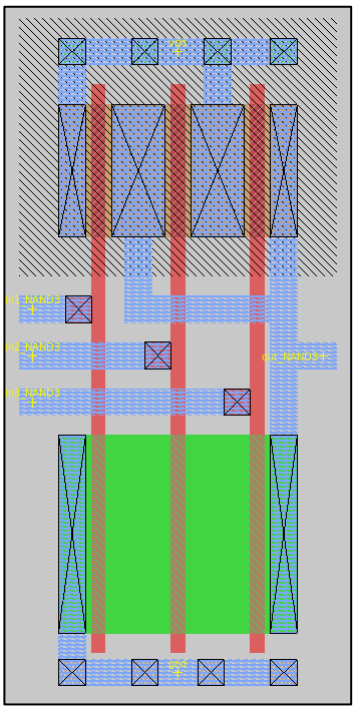
7.1 NOT Gate



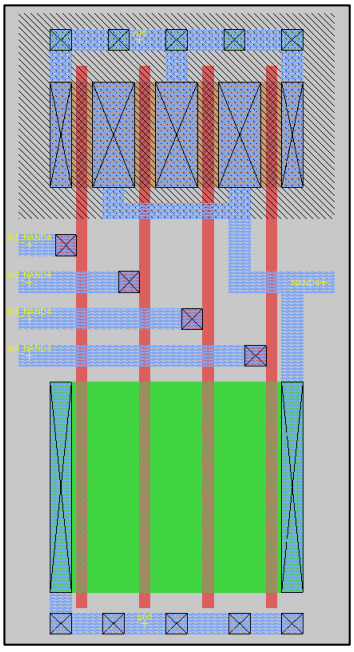
7.2 2-Input NAND



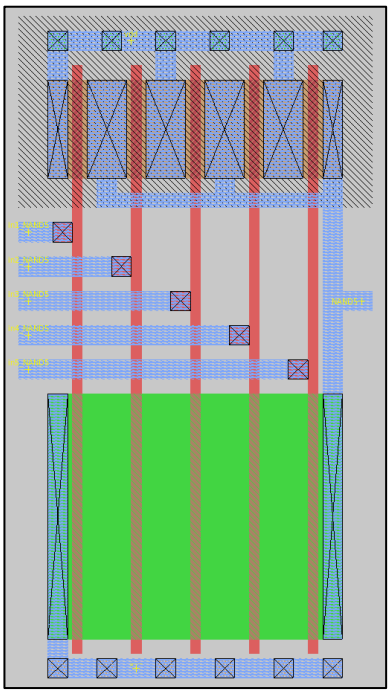
7.3 3-Input NAND



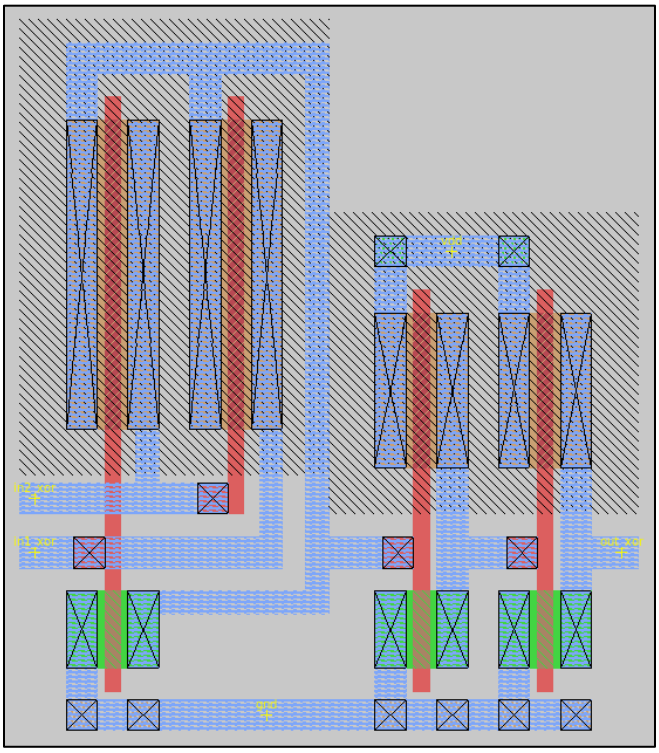
7.4 4-Input NAND



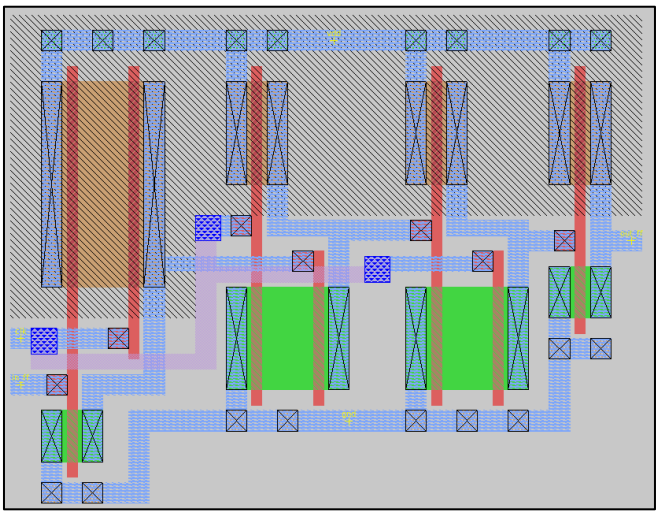
7.5 5-Input NAND



7.6 XOR Gate



7.7 D-Flip-Flop



VIII. FULL CIRCUIT INTEGRATION USING NGSPICE

In this section, the various blocks designed and validated in previous steps—XOR gates, NAND gates, D-flip-flops, and the CLA logic—are integrated to construct the complete Carry Look-Ahead Adder (CLA) circuit. The integrated design is simulated using NGSPICE to verify its functionality, measure the worst-case delay, and determine the maximum clock speed for correct operation.

8.1 Full Circuit Netlist

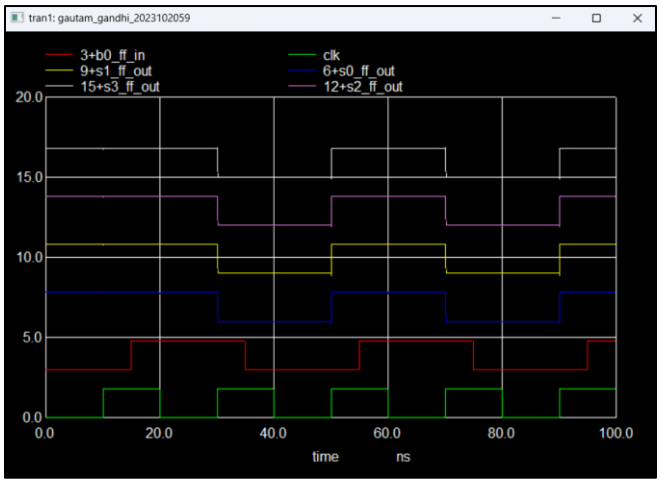
The netlist for the full 4-bit CLA adder circuit is constructed by interconnecting the following blocks:

- **Propagate and Generate Block**:
  - XOR gates $p_i = a_i \oplus b_i$ and NAND gates for $g_i = a_i \cdot b_i$.
- **Carry Generation Block**:
  - Hierarchical logic to compute carries (C_1, C_2, C_3, C_4) using recursive NAND-based carry expressions.
- **Sum Generation Block**:
  - XOR gates to compute sums $S_i = p_i \oplus C_i$.
- **Flip-Flops**:
  - Positive edge-triggered D-flip-flops implemented using TSPC technology for data synchronization.

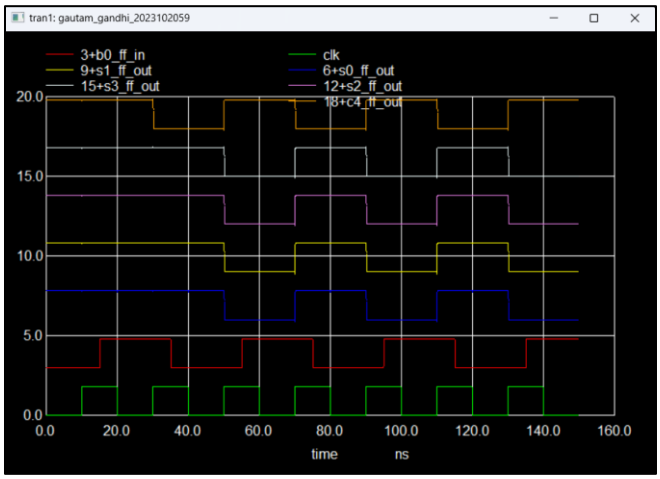
8.2 Functional Verification

On the first rising edge of the clock, inputs are taken and on the next rising edge, outputs are displayed.
The netlist was tested for different input combinations. The NGSPICE simulations verified the correctness of the CLA adder.

- For a = 0 0 0 0, and b = pulse pulse pulse pulse, and c_0 = 0 (these are inputs to the flip-flops), the outputs are obtained as:



- For a = 1 1 1 1, b = 0 0 0 0 pulse, c_0 = 0, the output waveform is:



- The sum and carry outputs match the theoretical expectations for all test cases and the waveforms confirm the correctness of these outputs.

8.3 Worst Case Delay

- The delay is measured from the input transition to the stabilization of the final output (C_4 or S_3).
- Worst-case scenarios involve the longest carry propagation path.
- The worst-case delay can be obtained by keeping a = 1111 and b = 0 0 0 0 pulse. This will change C_4 from 1 to 0 and 0 to 1.

Using this technique, the worst-case delay was measured to be **0.495 ns.**

8.4 Maximum Clock Speed

The maximum clock speed is determined by the time period required to accommodate the worst-case delay, setup time, and hold time:

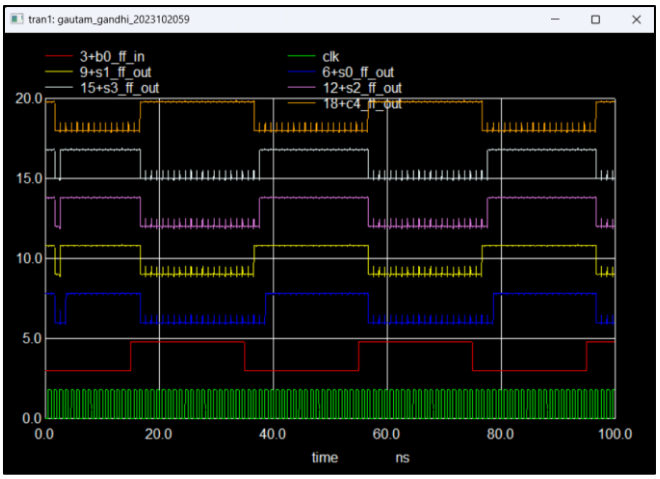
$$T_{min} = t_{clk-to-Q} + t_{worst-case} + t_{setup}$$

Substituting values:

$$T_{min} = 93\,ps + 495\,ps + 29\,ps = 617\,ps$$

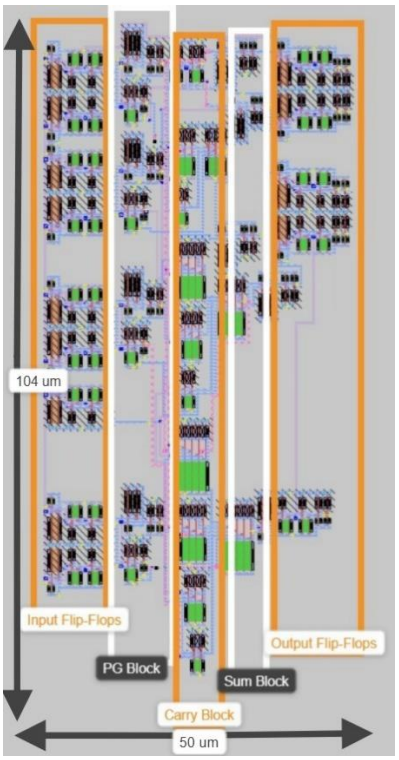
Thus, the **Maximum Clock Frequency** is:

$$f_{max} = \frac{1}{T_{min}} = \mathbf{1.62\,GHz}$$



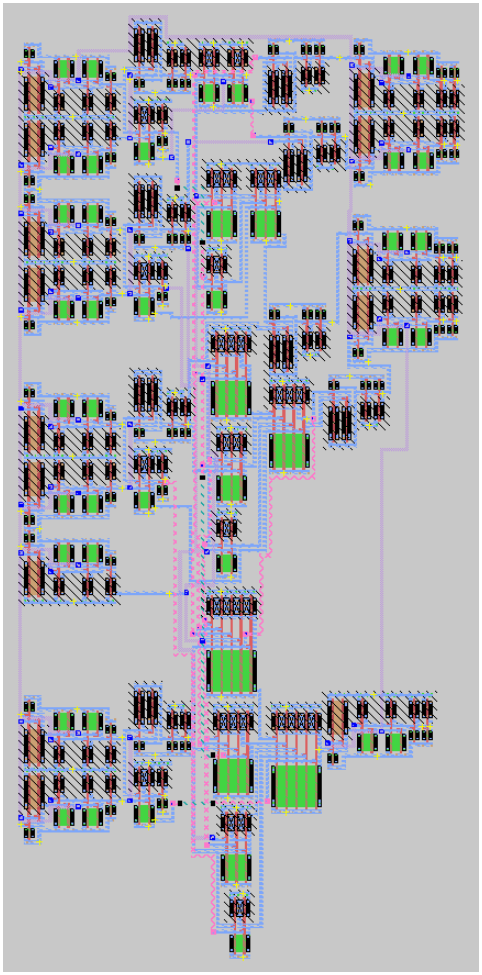
IX. FLOOR PLAN AND PITCH ANALYSIS FOR THE CLA ADDER LAYOUT

The following outlines the floor plan for the complete layout of the Carry Look-Ahead Adder (CLA) circuit, created using MAGIC layout editor. The floor plan organizes the circuit components systematically to optimize area, reduce routing complexity, and maintain regularity in the layout.



X. COMPLETE CIRCUIT LAYOUT DESIGN

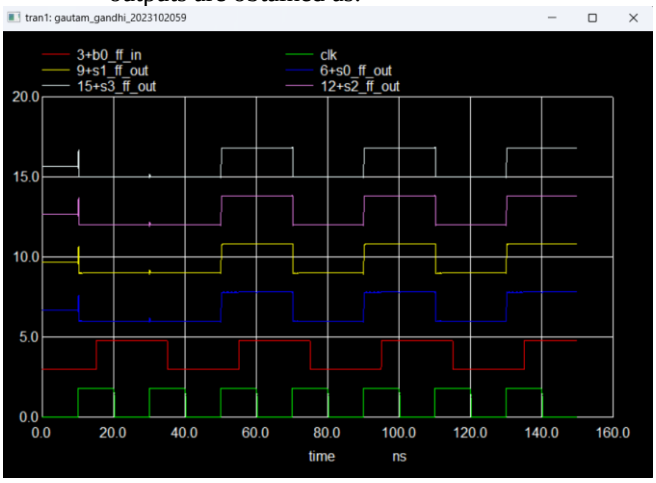
In this section, the various blocks designed and validated in previous steps—XOR gates, NAND gates, D-flip-flops, and the CLA logic—are integrated to construct the complete Carry Look-Ahead Adder (CLA) circuit using MAGIC.



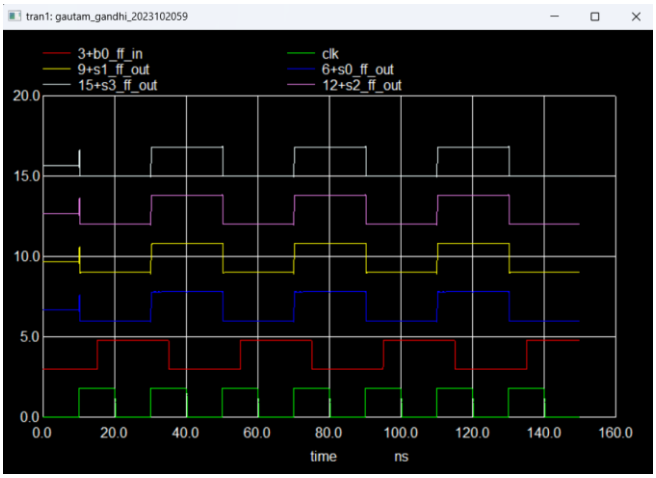
Functional Verification

On the first rising edge of the clock, inputs are taken and on the next rising edge, outputs are displayed.
The netlist was tested for different input combinations. The NGSPICE simulations verified the correctness of the CLA adder.

- For a = 0 0 0 0, and b = pulse pulse pulse pulse, and $c_0 = 0$ (these are inputs to the flip-flops), the outputs are obtained as:



- For a = 1 1 1 1, b = 0 0 0 pulse, $c_0 = 0$, the output waveform is:



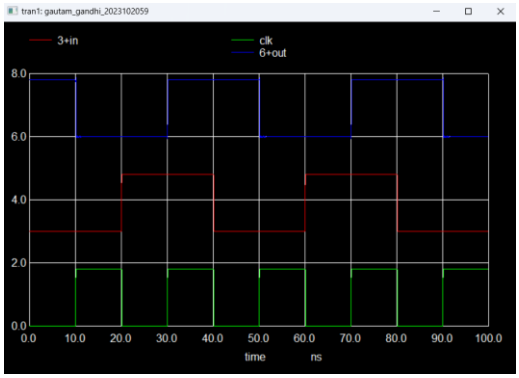
- The sum and carry outputs match the theoretical expectations for all test cases and the waveforms confirm the correctness of these outputs.

XI. POST LAYOUT SIMULATIONS

- The layout was processed in MAGIC to extract the parasitic RC components and create a post-layout SPICE netlist.
- The netlist includes additional parasitic capacitances and resistances associated with:
  - Transistor gates, drains, and sources.
  - Metal interconnects and vias.
- The extracted netlist was simulated in NGSPICE with the same input patterns used for the schematic simulations.

11.1 D-Flip-Flop Testing

The flip-flop is tested by giving clock and input pulses having different time periods. The following plot confirms the correct functionality of our positive edge triggered D-Flip-Flop.



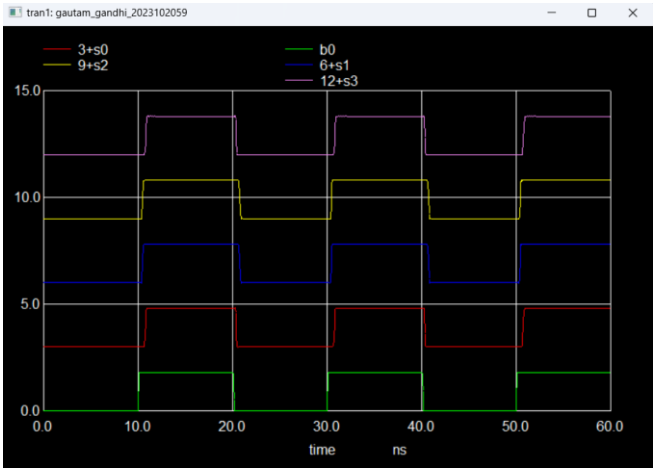
Using the same methods used earlier, the **setup time (35 ps)**, **hold time (19 ps)**, and **clock-to-Q delay (108 ps)** were calculated.

11.2 CLA Adder Testing

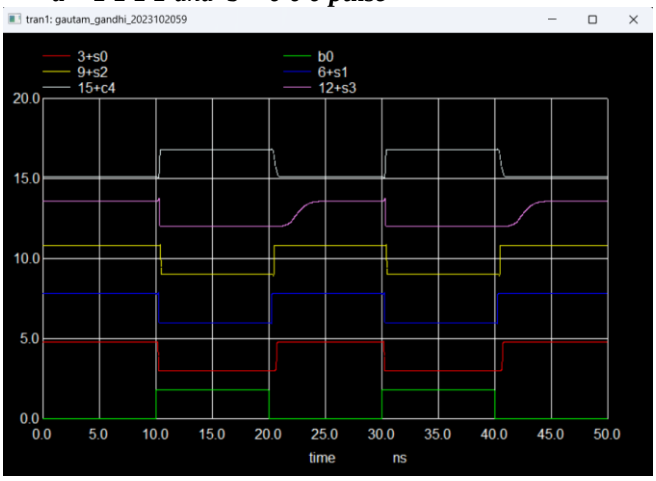
The CLA Adder is tested for different test cases:

- ***a = 0 0 0 0*** *and* ***b = pulse pulse pulse pulse***

The expected output is S = b, and C_4=*0.*



- ***a = 1 1 1 1*** *and* ***b = 0 0 0 0 pulse***



Conclusion:

The above plots confirm the correct functioning of the CLA Adder.

11.3 Worst Case Delay

The worst-case delay of the circuit was measured to be 0.586ns.

11.4 Maximum Clock Frequency

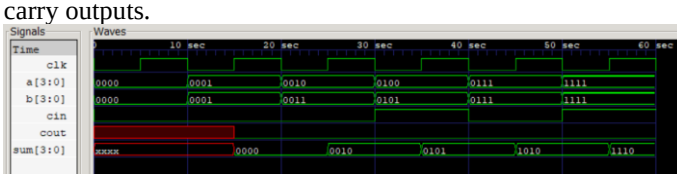
The max. clock frequency was calculated as 1.37 GHz.

11.1 Pre-Layout vs Post Layout Comparison

Time	Pre-Layout	Post-Layout
Setup Time	29 ps	35 ps
Hold Time	12 ps	19 ps
Clock-to-Q	93 ps	108ps
Worst Case Delay	495 ps	586 ps
Max. Clock Frequency	1.62 GHz	1.37 GHz

XII. VERILOG IMPLEMENTATION

This section provides the structural Verilog description of the Carry Look-Ahead Adder (CLA) circuit and demonstrates its functionality through simulation. The correctness of the design is validated by observing the waveforms for sum and carry outputs.



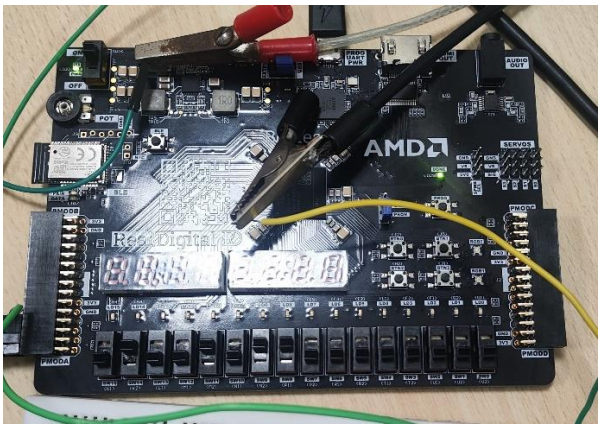
On the first rising edge of the clock, inputs are taken and on the next rising edge, outputs are displayed.

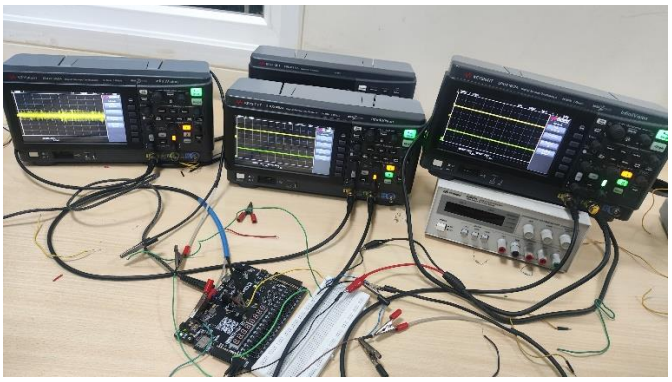
XIII. FPGA IMPLEMENTATION

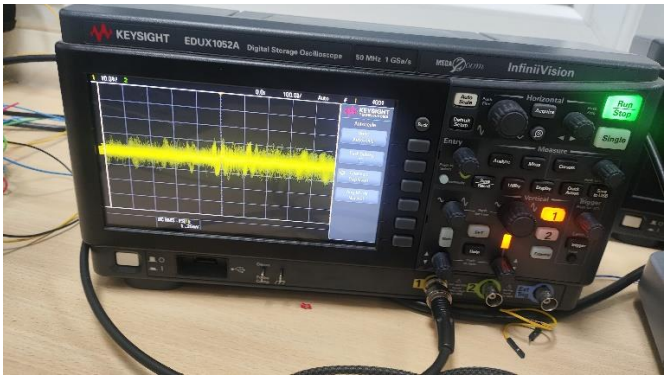
We provide the design file to the Boolean board along with the appropriate constraints file. The clock button is pressed twice to get the output at second POSITIVE edge. The inputs are labelled as shown:

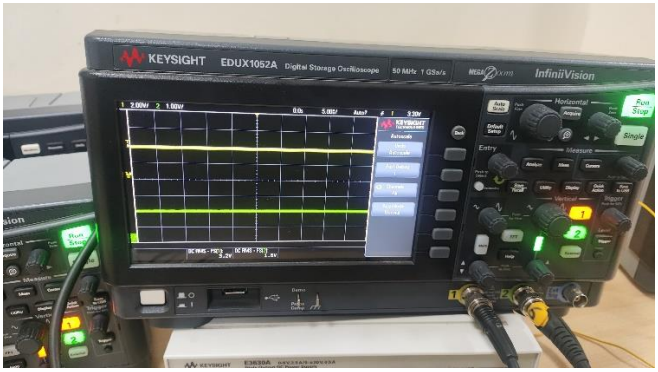
Oscilloscope Waveforms

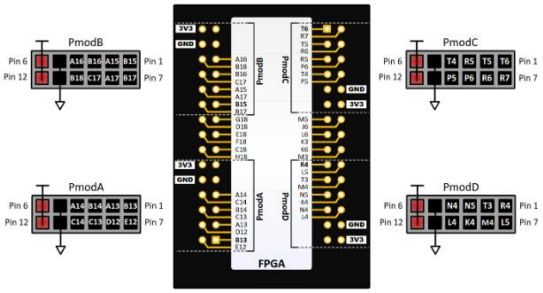
The input was given as a=0100 and b=1011, c0=0.
The output was observed as c4=0 and s=1111.











References

A variety of resources, articles, and discussions were utilized in identifying all the requested solutions. I would like to extend my gratitude to the professor and the teaching assistants for making the project and course productive.

[1] Yu-Ting Pai and Yu-Kumg Chen, ”The fastest carry lookahead adder,” Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1409882&isnumber=30560

[2] TSPC Logic, ”Circuit for all seasons.” Available at: https://ieeexplore.ieee.org/document/7743122

[3] New Efficient Design for XOR Function on the Transistor Level. Available at: https://doi.org/10.1063/1.3526229

[4] M. Morris Mano, ”Digital Design.” Available at: https://archive.org/details/m.-morris-mano-morris-m-mano-digital-design-3r-book-zz.org