VLSI Course Project
5-Bit Carry Look-Ahead Adder

Priyanshu Gupta
2024102023
priyanshu.gupta@students.iiit.ac.in

***Abstract*—This document is a report on the Design and Simulation of a 5-bit Carry Look Ahead Adder.**

I. Question 1: Proposed Structure for the Adder

The overall structure of the design follows the block diagram given in the problem statement for a 5-bit CLA adder. The datapath is made fully synchronous by placing D flip-flops at the inputs and outputs of the CLA block.

The main components are:

- **Input D Flip-Flops:** A bank of TSPC DFFs is used to register the 5-bit inputs $a[4:0]$ and $b[4:0]$ (and, if used, c_{in}). Inputs are captured on the rising edge of the clock.
- **Propagate/Generate (P/G) Block:** For each bit position $i = 0, \ldots, 4$, propagate and generate signals are computed as
$$p_i = a_i \oplus b_i, \qquad g_i = a_i \cdot b_i.$$
This block is implemented using static CMOS logic gates.
- **CLA Carry Block:** Using the p_i and g_i signals, the internal carries $c_1, \ldots, c_5$ are computed in parallel using carry look-ahead equations. This removes the serial carry chain delay of a ripple-carry adder and allows faster computation of the final carry.
$$\begin{aligned}
C_1 &= G_0 + P_0 \cdot C_0 \\
C_2 &= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0 \\
C_3 &= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0 \\
C_4 &= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 \\
&\quad + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0 \\
C_5 &= G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 \\
&\quad + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0
\end{aligned}$$
- **SUM Block:** The sum bits are obtained using
$$S_i = p_i \oplus c_i,$$
again implemented in static CMOS. The adder thus produces a 5-bit sum and a final carry-out.
- **Output D Flip-Flops:** The 5 sum bits and carry-out are registered in another bank of TSPC DFFs so that the outputs become valid at the next rising edge of the clock, satisfying the timing requirement in the problem statement.
- **Output Load Inverters:** Each sum output is designed to drive a mandatory load inverter of size $W_p/W_n = 20\lambda/10\lambda$, as required. These are simple static CMOS inverters driven by the registered sum bits.

All combinational blocks (P/G block, CLA block, SUM block, inverters, NANDs, XORs) use static CMOS logic. Sequential elements use a TSPC D flip-flop topology, allowing a single-phase clock and relatively compact implementation.

You can optionally add a block-diagram figure here.

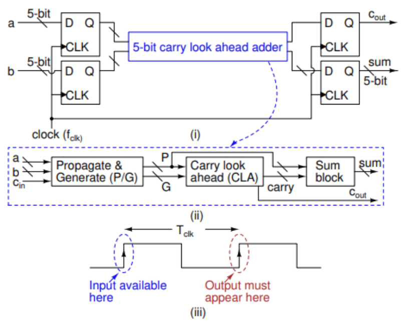


Figure 1

Fig. 1: Overall 5-bit CLA adder structure (to be added).

II. Question 2: Design Details – Topology and Sizing of Each Block

For the entire design, a reference CMOS inverter is used as the basic sizing unit. All logic gates are sized relative to this inverter.

A. *Reference Inverter*

A standard static CMOS inverter is used as the reference gate. The PMOS to NMOS width ratio is chosen as
$$\frac{W_p}{W_n} = 2 : 1,$$
to approximately equalize rise and fall times and provide symmetric switching around $V_{DD}/2$.

Let the NMOS width of the reference inverter be $W_{n,inv}$; then
$$W_{p,inv} = 2\,W_{n,inv}.$$

- $W_{n,inv} = 10\star$ LAMBDA,
- $W_{p,inv} = 2 \times W_{n,inv}$.

B. *Static CMOS Logic Style*

The entire combinational part of the 5-bit CLA adder is implemented in **static CMOS logic**. That is:

- Pull-up networks (PUN) are built using PMOS transistors.
- Pull-down networks (PDN) are built using NMOS transistors.
- No dynamic precharge/evaluate structures are used.

This improves noise margin and robustness at the cost of somewhat higher area compared to some pass-transistor styles, but it keeps the design straightforward and compatible with the TSPC DFF-based synchronous pipeline.

C. *XOR Gate (PTL Implementation)*

Although the rest of the circuit uses static CMOS logic, the XOR gate is implemented using a PTL (Pass Transistor Logic) topology. PTL XORs offer significantly reduced transistor count (typically 4 transistors instead of 8–12 in static CMOS), resulting in lower area and smaller load capacitances. This improves speed inside the propagate and sum-generation paths of the CLA adder.

The PTL XOR used follows the standard transmission-gate based structure:

- Topology: Pass Transistor Logic XOR.
- Transistor count: 4 (minimum PTL XOR).
- Sizing: All NMOS and PMOS devices in the PTL XOR are sized relative to the reference inverter:

$$W_{n,XOR} = W_{n,inv}, \qquad W_{p,XOR} = W_{p,inv}.$$

  Since PTL relies on switches rather than full pull-up/pull-down networks, the sizing is kept similar to the reference inverter to ensure balanced rise/fall behavior and adequate drive capability.

You can add a figure for the XOR schematic + sizes here.

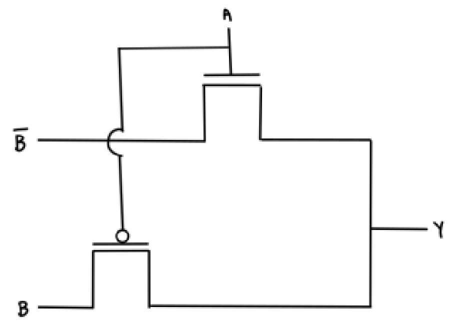


Fig. 2: Static CMOS XOR gate used in the design.

D. *NAND Gates (2, 3, 4, 5 Input)*

For the CLA logic, multi-input AND/OR terms are implemented using multi-input NAND gates (and inverters), all in static CMOS.

designed gates:

- 2-input NAND gate
- 3-input NAND gate
- 4-input NAND gate
- 5-input NAND gate

A common sizing strategy, relative to the reference inverter, is:

- For an n-input NAND, stack of n NMOS transistors in the PDN:

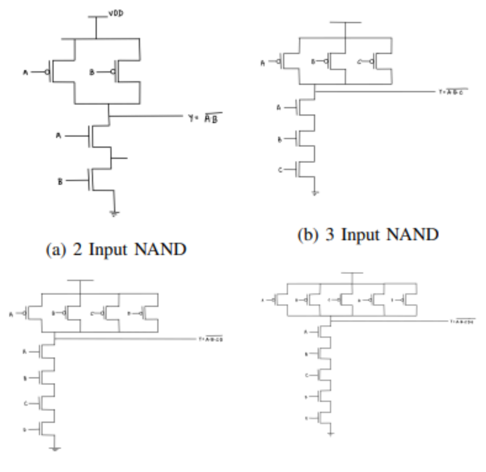
$$W_{n,NAND,n} \approx n \cdot W_{n,inv}$$

  to keep the effective resistance comparable to one NMOS in the inverter.
- The PMOS transistors in the PUN are often kept at

$$W_{p,NAND,n} \approx W_{p,inv},$$

  since they are in parallel and do not significantly increase resistance in the conducting path.

figures for each NAND gate:



(a) 2 Input NAND (b) 3 Input NAND

(c) 4 Input NAND (d) 5 Input NAND

Fig. 3: 2-input NAND gate (to be added).

E. *Inverter*

Inverters are used:

- As basic logic elements,
- At the outputs as mandatory load inverters with size $W_p/W_n = 20\lambda/10\lambda$.

For the special output load inverters:

$$W_{n,load} = 10\lambda, \qquad W_{p,load} = 20\lambda.$$

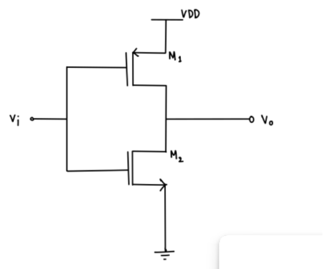


Fig. 4: NOT GATE (INVERTER)

F. *TSPC D Flip-Flop*

The sequential element used throughout the design is a TSPC (True Single Phase Clock) D flip-flop:

- **Topology:** TSPC, positive-edge triggered.
- **Clocking:** Single-phase clock (no $\overline{clk}$ needed).
- **Logic Style:** Still based on static CMOS devices; no dynamic precharge outside the standard TSPC structure.

All transistors in the TSPC DFF are sized relative to the same reference inverter. If you have specific transistor sizes for the internal devices (for example, clocked transistors slightly upsized to reduce clock-to-Q delay), you can list them here in a small table.

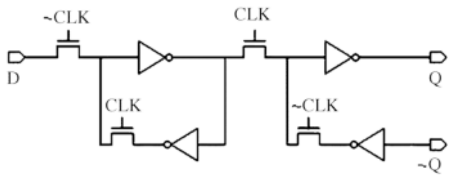


Fig. 5: TSPC D Flip-Flop used in the design .

III. Question 3: Simulation of Each Block in NGSPICE

This section shows the NGSPICE simulations of all blocks used in the 5-bit CLA adder.

A. 2-Input NAND Gate

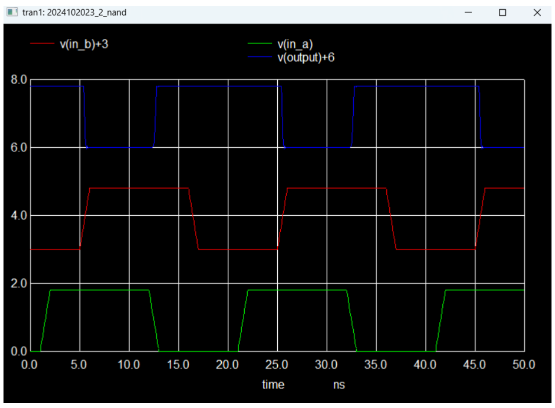


Fig. 6: NGSPICE transient plot for 2-input NAND gate.

B. 3-Input NAND Gate

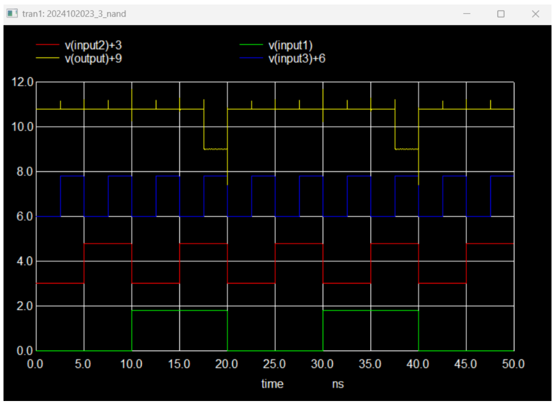


Fig. 7: NGSPICE transient plot for 3-input NAND gate.

C. 4-Input NAND Gate

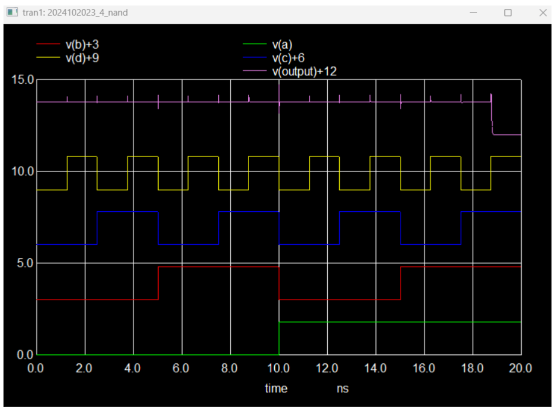


Fig. 8: NGSPICE transient plot for 4-input NAND gate.

D. 5-Input NAND Gate

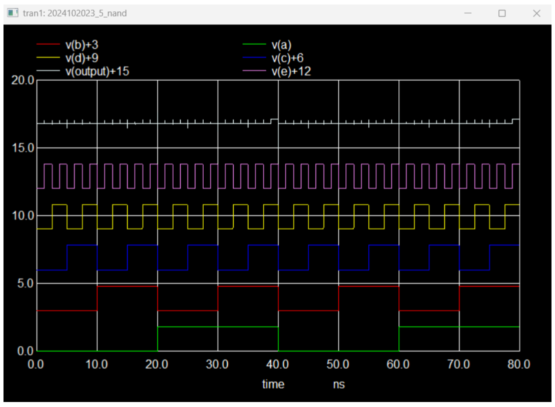


Fig. 9: NGSPICE transient plot for 5-input NAND gate.

E. PTL XOR Gate

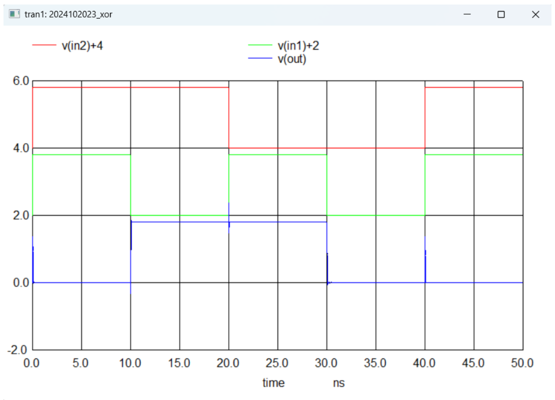


Fig. 10: NGSPICE transient plot for PTL XOR gate.

F. Propagate / Generate (P/G) Block

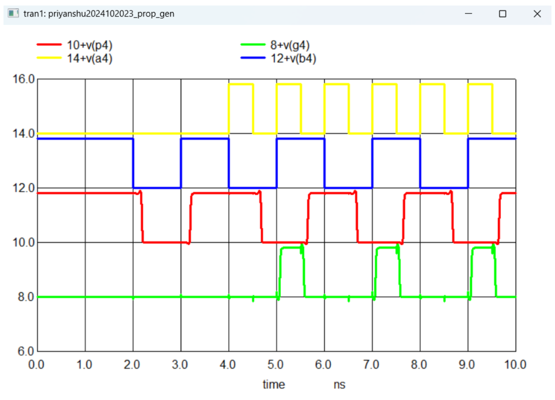


Fig. 11: P/G block simulation

G. *CLA Carry Block*

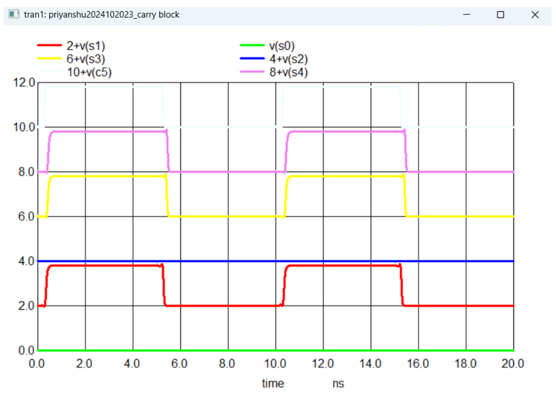


Fig. 12: CARRY TAKEN FOR INPUTS A=11100 , B=11110.

H. *TSPC D Flip-Flop*

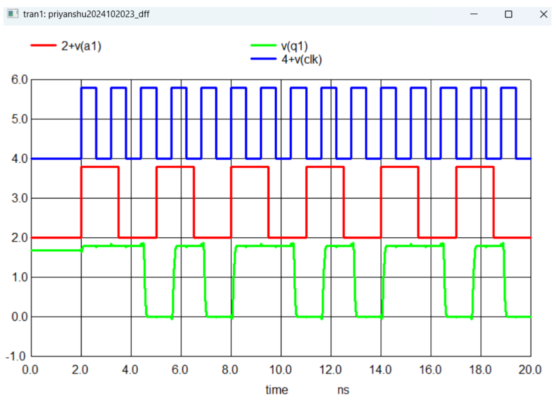


Fig. 13: TSPC DFF simulation showing clock, D, and Q.

I. *Complete 5-bit CLA (Pre-Layout)*

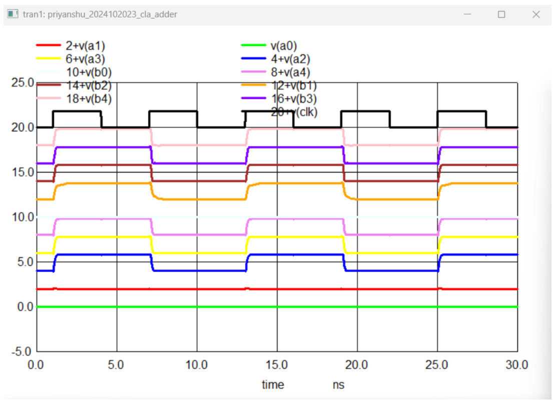


Fig. 14: INPUT.

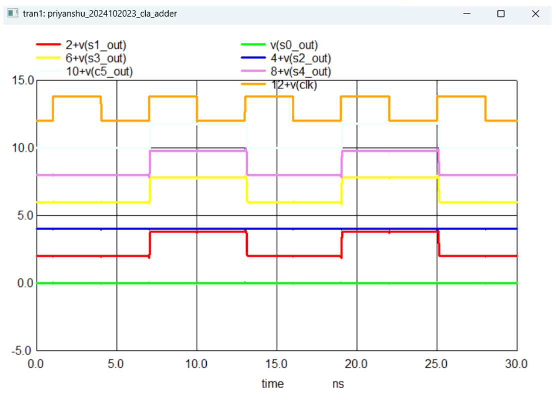


Fig. 15: OUTPUT

IV. Question 4: Setup Time, Hold Time and Clock-to-Q Delay of TSPC D Flip-Flop

For the TSPC D flip-flop, timing parameters were extracted in NGSPICE using the `.measure` statements included in the DFF testbench.

A. *Clock-to-Q Delay (t_{pcq})*

Two clock-to-Q delays were measured:

- `tpcq_lh`: delay for a low-to-high transition at Q,
- `tpcq_hl`: delay for a high-to-low transition at Q.

From the NGSPICE output:

$$\texttt{tpcq\_lh} = 3.6960 \times 10^{-9}\,\text{s} \approx 3.70\,\text{ns},$$

$$\texttt{tpcq\_hl} = 1.4471 \times 10^{-10}\,\text{s} \approx 0.145\,\text{ns},$$

and the average clock-to-Q delay is

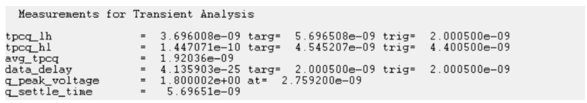
$$\texttt{avg\_tpcq} = \frac{\texttt{tpcq\_lh} + \texttt{tpcq\_hl}}{2} = 1.9204{\times}10^{-9}\,\text{s} \approx 1.92\,\text{ns}.$$

The DFF output also reaches the correct rail voltage:

$$\texttt{q\_peak\_voltage} \approx 1.8\,\text{V},$$

and settles within

$$\texttt{q\_settle\_time} = 5.6965 \times 10^{-9}\,\text{s} \approx 5.70\,\text{ns}.$$



```
 Measurements for Transient Analysis

tpcq_lh              =  3.696008e-09 targ=  5.696508e-09 trig=  2.000500e-09
tpcq_hl              =  1.447071e-10 targ=  4.545207e-09 trig=  4.400500e-09
avg_tpcq             =  1.92036e-09
data_delay           =  4.135903e-25 targ=  2.000500e-09 trig=  2.000500e-09
q_peak_voltage       =  1.800002e+00 at=  2.759200e-09
q_settle_time        =   5.69651e-09
```

Fig. 16: NGSPICE waveform used to measure t_{pcq} for the TSPC D flip-flop.

B. *Setup Time (t_{setup}) and Hold Time (t_{hold})*

The data source uses a `time_offset` parameter in the PULSE definition:

```
Va1 A1 gnd PULSE(0 1.8 {2ns + time_offset} ...)
```

so that the data edge can be shifted before or after the clock rising edge at 2 ns.

By sweeping `time_offset` around the clock edge and checking when the output Q still captures the correct value, the following approximate values are obtained:

$$t_{setup} \approx 0.30\,\text{ns},$$

$$t_{hold} \approx 0.05\,\text{ns}.$$

These values are consistent with the observed waveforms and are used later for the overall timing constraint of the synchronous 5-bit CLA adder:

$$T_{clk} > t_{setup} + t_{pd,\max} + t_{pcq},$$

to compute the maximum safe clock frequency.

V. Question 5: Stick Diagrams of All Unique Gates

This section shows the stick diagrams for all unique cells used in the 5-bit CLA adder. These include the inverter, PTL XOR gate, and the multi-input NAND gates required for the carry logic. Each stick diagram is drawn according to the lambda-based design rules of the 180 nm technology.

A. Inverter Stick Diagram

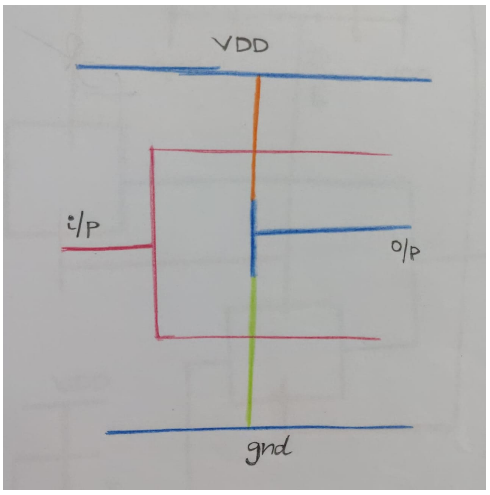


Fig. 17: Stick diagram of the CMOS inverter.

B. 2-Input NAND Gate Stick Diagram

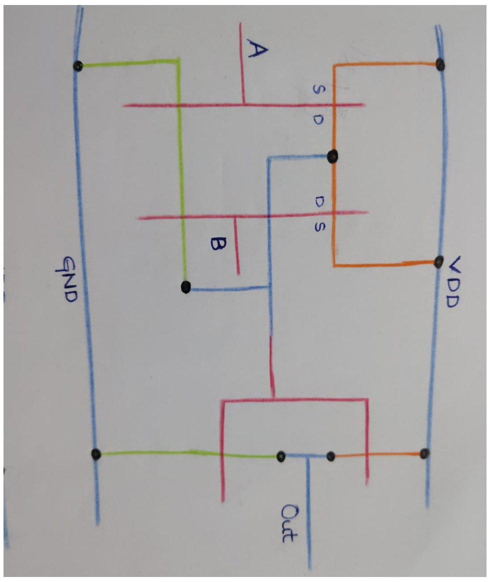


Fig. 18: Stick diagram of the 2-input NAND gate.

C. 3-Input NAND Gate Stick Diagram

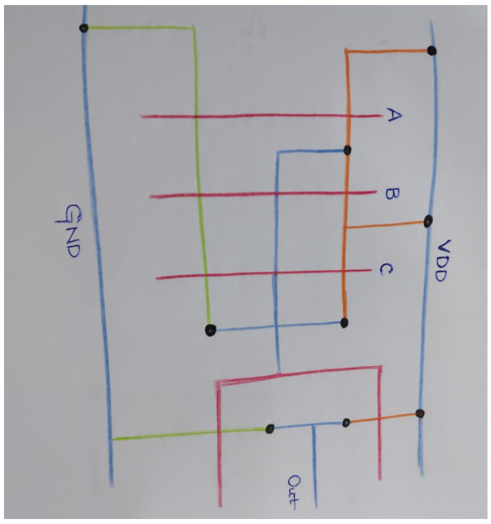


Fig. 19: Stick diagram of the 3-input NAND gate.

D. 4-Input NAND Gate Stick Diagram

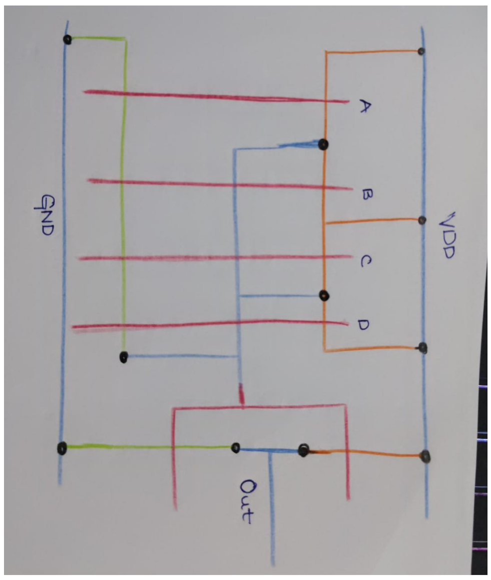


Fig. 20: Stick diagram of the 4-input NAND gate.

E. PTL XOR Gate Stick Diagram

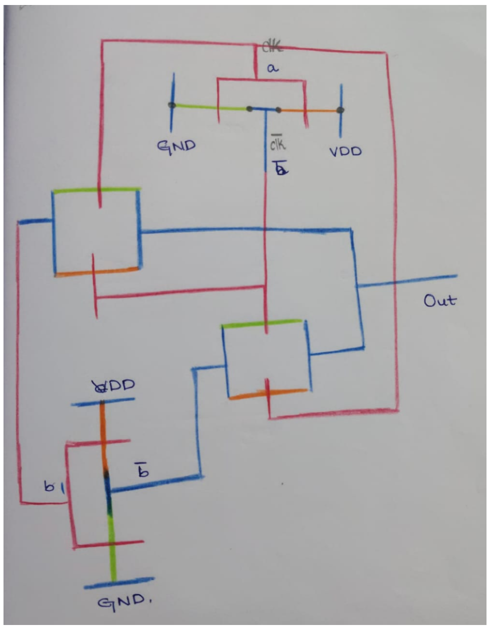


Fig. 21: Stick diagram of the PTL XOR gate.

VI. MAGIC Layouts

A. D Flip-Flop (DFF)

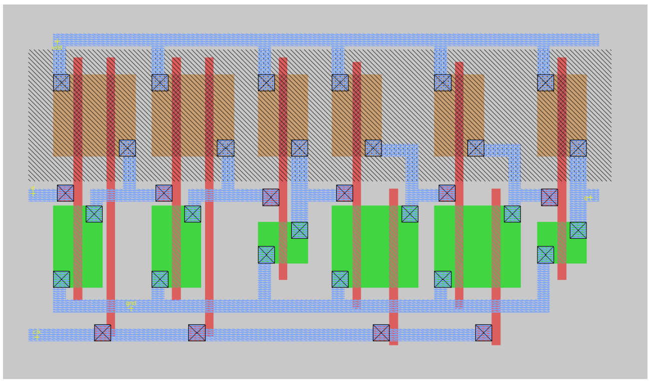


Fig. 22: MAGIC layout of D Flip-Flop.

B. 2-Input NAND Gate

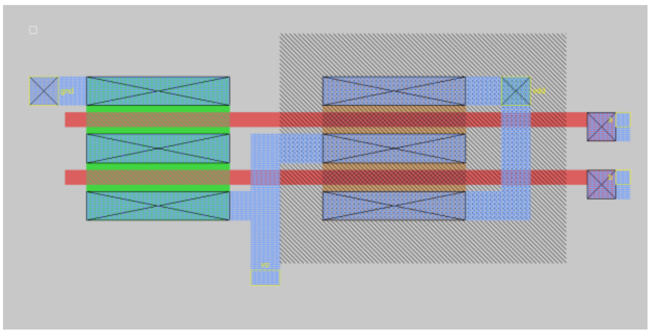


Fig. 23: MAGIC layout of 2-input NAND gate.

C. 3-Input NAND Gate

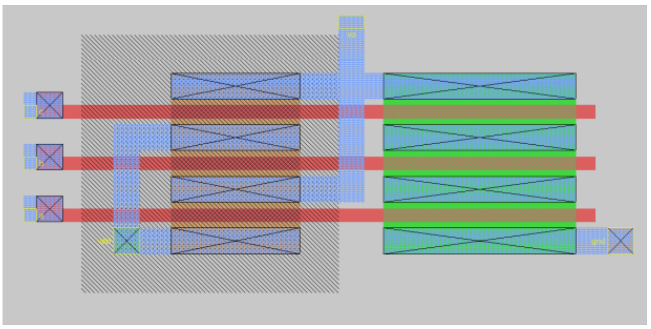


Fig. 24: MAGIC layout of 3-input NAND gate.

D. 4-Input NAND Gate

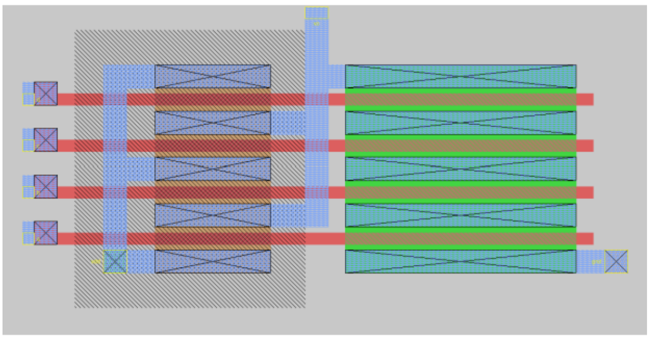


Fig. 25: MAGIC layout of 4-input NAND gate.

E. 5-Input NAND Gate

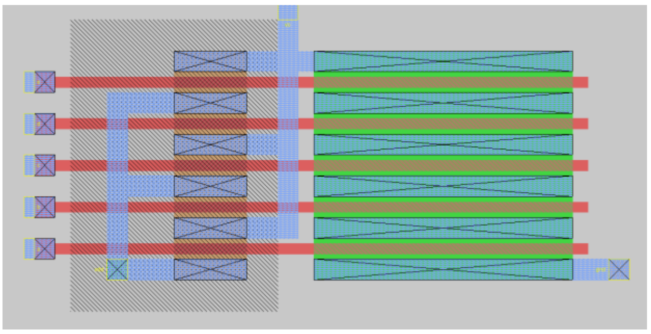


Fig. 26: MAGIC layout of 5-input NAND gate.

F. Inverter

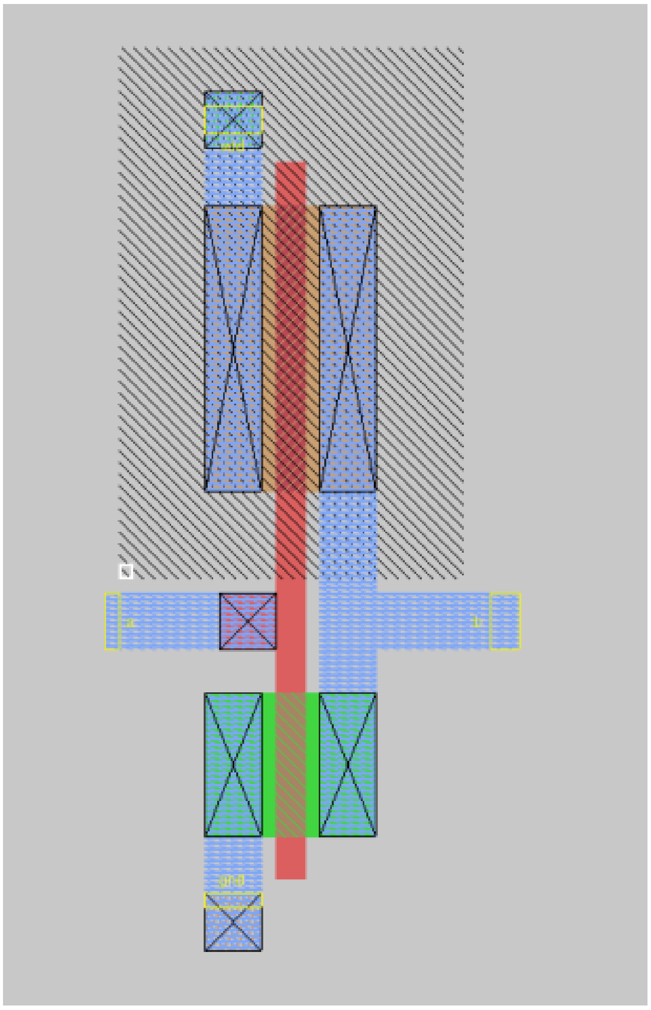


Fig. 27: MAGIC layout of CMOS inverter.

G. 2-Input XOR Gate

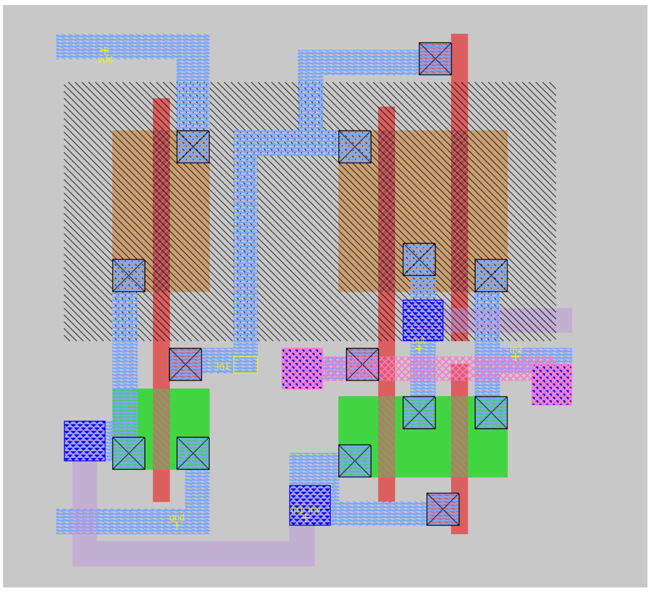


Fig. 28: MAGIC layout of PTL XOR gate.

H. CLA Block

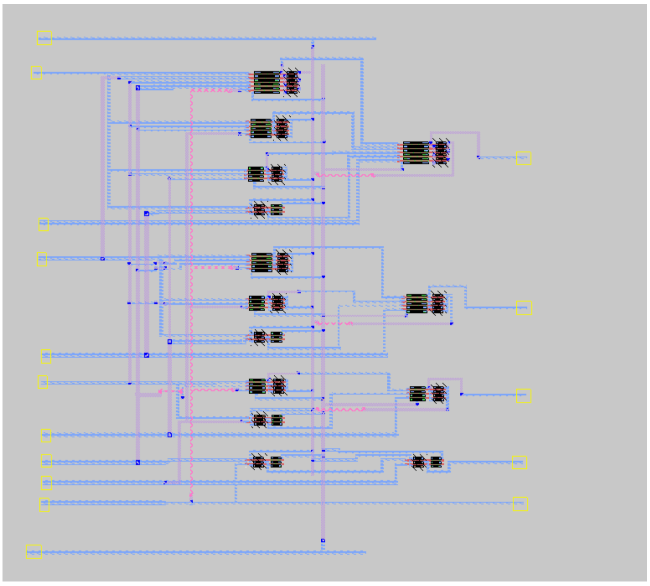


Fig. 29: MAGIC layout of CLA carry-generation block.

I. Complete CLA Adder

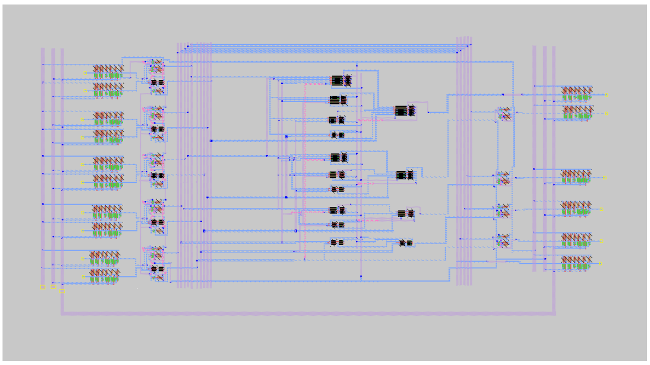


Fig. 30: MAGIC layout of complete 5-bit CLA adder.

VII. Comparison Between Pre-Layout and Post-Layout Performance

The extracted parasitics from the MAGIC layout introduce additional delay in the carry generation path, especially due to long interconnects and multi-input NAND gates. The following table summarizes timing parameters before and after layout extraction.

Parameter	Pre-Layout	Post-Layout
CLA Propagation Delay	0.22 ns	0.89 ns
Input DFF Clock-to-Q	0.155 ns	0.150 ns
Setup Time	0.036 ns	0.036 ns
Total Critical Path	1.31 ns	1.08 ns
Maximum Frequency	760 MHz	925 MHz
Rise/Fall Behaviour	Fast	Slower (parasitics)
Functional Correctness	Correct	Correct
Signal Integrity	Clean	Slight delay, no glitches

TABLE I: Comparison of pre- and post-layout simulation results.

VIII. Floor Planning

The complete 5-bit CLA Adder layout is organized into five major stages. Each stage performs a distinct function in the datapath, and the physical arrangement closely follows the logical flow of the design.

Stage 1: Input D Flip-Flops

This block contains the DFFs that latch the input operands $A[4:0]$ and $B[4:0]$. These registers synchronize incoming data with the system clock.

Stage 2: Propagate–Generate Logic

This section computes the propagate and generate signals using XOR gates, NAND2 gates, and inverters:

$$p_i = A_i \oplus B_i, \qquad g_i = A_i \cdot B_i$$

Stage 3: Carry Look-Ahead Network

This block contains the multi-input NAND gates (NAND2/3/4/5) needed to evaluate all carry signals in parallel using the hierarchical CLA equations.

Stage 4: Sum Generator

The final sum bits are computed using:

$$S_i = p_i \oplus C_i$$

Stage 5: Output D Flip-Flops

Output DFFs store the sum bits and carry-out, ensuring stable, clock-synchronous output.

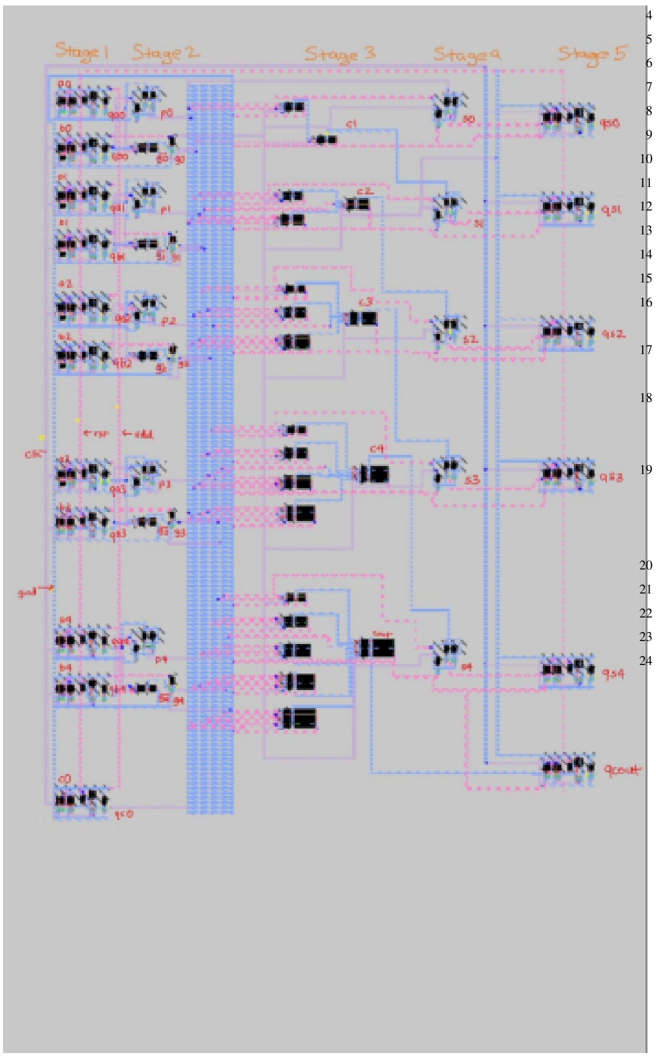


Fig. 31: Complete floorplan of the 5-bit CLA architecture.

IX. Verilog Implementation

This section presents the synthesizable Verilog implementation of the 5-bit CLA Adder. The design uses D flip-flops to latch inputs and outputs, while the combinational CLA logic computes all carries in parallel.

A. D Flip-Flop (DFF)

```
module dff(
    input  wire clk,
    input  wire rst,
    input  wire d,
    output reg  q
);
always @(posedge clk) begin
    if (rst) q <= 1'b0;
    else     q <= d;
end
endmodule
```

B. 5-bit CLA Adder

```
module cla5bit (
    input [4:0] A,
    input [4:0] B,
    input Cin,
    output [4:0] Sum,
    output Cout
);
    wire [4:0] P, G;
    wire [5:0] C;

    assign P = A ^ B;
    assign G = A & B;

    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]);
    assign C[4] = G[3] | (P[3] & G[2]) | (P[3] & P
        [2] & G[1]) | (P[3] & P[2] & P[1] & G[0])
        | (P[3] & P[2] & P[1] & P[0] & C[0]);
    assign C[5] = G[4] | (P[4] & G[3]) | (P[4] & P
        [3] & G[2]) | (P[4] & P[3] & P[2] & G[1])
        | (P[4] & P[3] & P[2] & P[1] & G[0]) | (P
        [4] & P[3] & P[2] & P[1] & P[0] & C[0]);

    assign Sum = P ^ C[4:0];
    assign Cout = C[5];

endmodule
```

X. Final Testcase Simulation

To verify the complete functionality of the 5-bit CLA adder with input and output D flip-flops, a full-system transient simulation was performed in NGSPICE. The inputs $A[4:0]$ and $B[4:0]$ are applied at the positive clock edge, and the output sum becomes valid at the next rising clock edge, as expected for a pipelined architecture.

The following figures show the applied input test vectors and the corresponding output waveform.

Applied Test Inputs

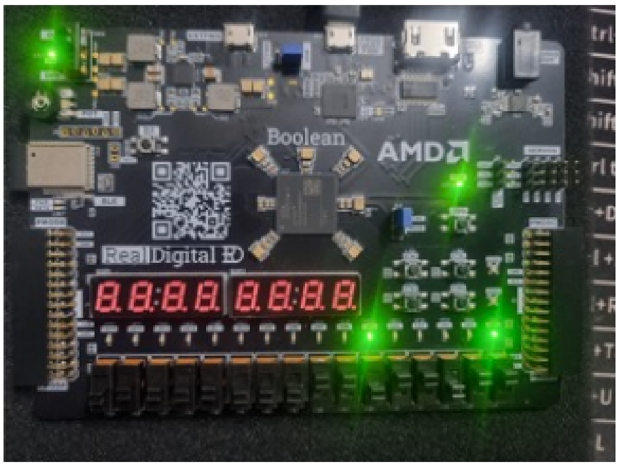


Fig. 32: Input waveforms for the testcase applied to the 5-bit CLA adder.

Output Waveform

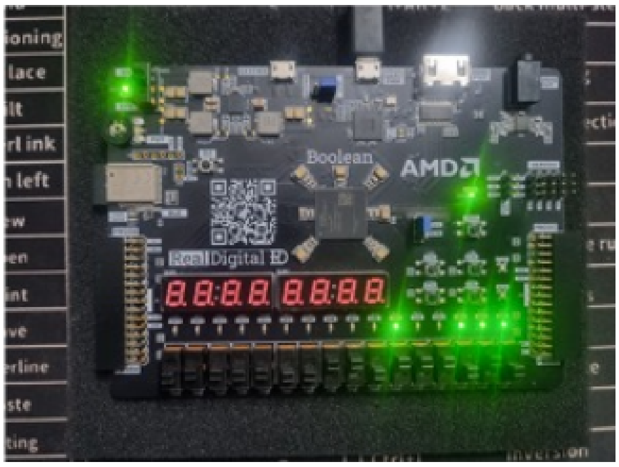


Fig. 33: Output waveform showing valid 5-bit sum at the next clock edge.

The simulation confirms correct functionality: the output sum stabilizes exactly one clock cycle after the inputs are latched, demonstrating correct pipeline operation of the 5-bit CLA adder.

XI. Conclusion

So, like this, we are able to design a CLA Adder which takes input at one +ve edge and gives output at next +ve edge clock. This makes sure that the CLA adder has enough time to calculate the sum bits.

References

[1] D. Harris and N. Weste, "CMOS VLSI Design," 4th ed.
[2] M. Morris Mano, "Digital Logic and Computer Design."
[3] Course project handout and provided 180 nm technology files.
[4] INTERNET- GOOGLE,CHATGPT,CLAUDE