5-Bit Carry Look-Ahead Adder: Design, Implementation and Analysis

Het Asit Selarka
Electronics and Communication Engineering
IIIT Hyderabad
Roll No. 2024102031
het.selarka@students.iiit.ac.in

***Abstract*—Addition is a fundamental operation in digital systems, significantly impacting the performance of arithmetic and logic units (ALUs), processors, and other digital circuits. Traditional ripple carry adders, while simple, suffer from high propagation delay due to the sequential dependency of carry generation. The Carry Look-Ahead Adder (CLA) addresses this limitation by employing a parallel approach to carry generation and propagation, resulting in reduced delay and enhanced performance for high-speed applications. This paper presents the complete design, implementation, and analysis of a 5-bit CLA adder using 180nm CMOS technology, including pre-layout and post-layout simulation results, FPGA implementation, and timing analysis with D flip-flops for synchronous operation.**

***Index Terms*—Carry Look-Ahead Adder, 5-bit Adder, TSPC D Flip-Flop, CMOS VLSI, Timing Analysis, FPGA And Vivado Implementation, MAGIC Layout**

I. Introduction

A 5-bit adder is a combinational digital circuit that accepts two unsigned 5-bit binary numbers $A = a_4a_3a_2a_1a_0$ and $B = b_4b_3b_2b_1b_0$ along with an optional input carry c_0, and produces a 5-bit sum output $S = s_4s_3s_2s_1s_0$ and a final carry-out c_5. At the most fundamental level, each bit position computes two internal quantities: the *propagate* signal $p_i = a_i \oplus b_i$ and the *generate* signal $g_i = a_i \cdot b_i$. These indicate whether a bit position will pass an incoming carry forward or generate a new one.

In a conventional ripple-carry adder (RCA), the carry-out from each bit must be computed before the next bit can begin its evaluation. As a result, the overall delay grows linearly with the number of bits: for an n-bit RCA, the delay is $O(n)$. For a 5-bit implementation, the carry must ripple through five full-adder stages, making it unsuitable for high-speed operation.

The Carry Look-Ahead Adder (CLA) significantly reduces this delay by *computing all carry signals in parallel*. The carry-out expressions are expanded algebraically in terms of p_i and g_i, allowing the circuit to determine c_1 through c_5 without waiting for sequential propagation. Because the carry logic is implemented through multi-level Boolean expressions instead of sequential dependency, the overall delay becomes a function of logic depth rather than bit-width. Thus, a 5-bit CLA achieves much lower latency and scales as $O(\log n)$ in practical implementations, resulting in substantial performance improvement over ripple-carry structures.

II. Architecture of 5-bit CLA Adder

A. Basic Structure

The 5-bit CLA adder consists of three main blocks as shown in Fig. 1:

- Propagate and Generate (P/G) Block
- Carry Look-Ahead Logic (CLA) Block
- Sum Generation Block

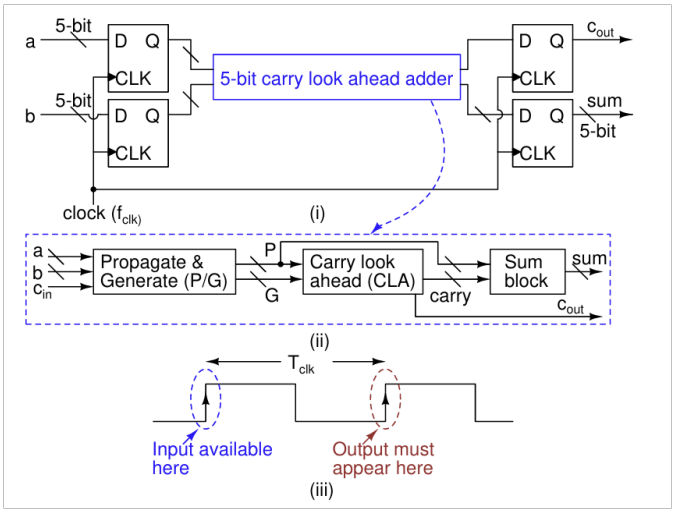


Fig. 1: Basic Circuit Diagram of Carry Look-Ahead Adder

Inputs are captured at one clock edge and outputs appear at the **next clock edge**.

For two 5-bit numbers $a_5a_4a_3a_2a_1$ and $b_5b_4b_3b_2b_1$, the propagate (p_i) and generate (g_i) signals for each bit position are defined as (for $i = 1, 2, 3, 4, 5$):

Propagate and Generate Signals: For each bit position, the propagate and generate signals are defined as:

$$p_i = a_i \oplus b_i, \qquad g_i = a_i \cdot b_i$$

Using these definitions, the carry-out for each bit can be written as:

$$c_{i+1} = g_i + (p_i \cdot c_i)$$

with the initial carry-in set to $c_0 = 0$.

Applying this relationship iteratively, the carry signals for all bit positions become:

$$c_1 = g_0$$

$$c_2 = (p_1 \cdot g_0) + g_1$$

$$c_3 = (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot g_1) + g_2$$

$$c_4 = (p_3 p_2 p_1 g_0) + (p_3 p_2 g_1) + (p_3 g_2) + g_3$$

$$c_5 = (p_4 p_3 p_2 p_1 g_0) + (p_4 p_3 p_2 g_1) + (p_4 p_3 g_2) + (p_4 g_3) + g_4$$

Finally, each sum bit is computed as:

$$S_i = p_i \oplus c_i$$

III. Design Details

A. *Static CMOS Realization Using NAND Universal Gates*

The complete 5-bit CLA architecture in this project is implemented using **static CMOS logic**, specifically utilizing **NAND-gate-based universal logic**. NAND is preferred in CMOS design because it provides:

- high noise margins,
- low static power consumption,
- highly regular layout structure,
- minimum number of transistors for universal logic implementation.

All major combinational blocks—propagate/generate (P/G) logic, carry look-ahead logic, and the sum generation logic—are built from NAND networks. This provides better transistor count efficiency, ease of layout design, and predictable timing behavior.

Since the design is fully static CMOS, every gate possesses strong pull-up and pull-down networks, ensuring full logic swings, glitch-free operation, and robust behavior across process, voltage, and temperature variations.Thereby, this logic proves to be more robust than Pass Transistor Logic.
The proposed 5-bit CLA adder uses a CMOS-based implementation with the following elements:

1) Inverter (INV): The inverter is the fundamental building block with:

- NMOS: $W_N = 1.8\mu m$, $L = 2\lambda = 0.18\mu m$
- PMOS: $W_P = 3.6\mu m$, $L = 2\lambda = 0.18\mu m$

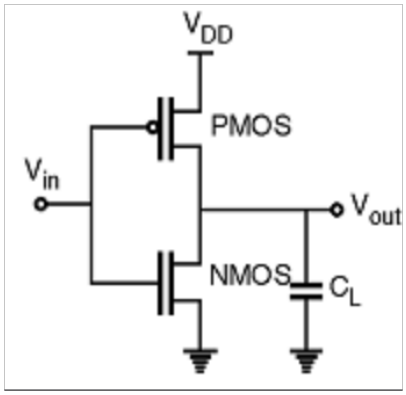


Fig. 2: Inverter

2) 2-Input NAND Gate: The 2-input NAND gate uses:

- Series NMOS: $W = 2 \times W_N = 3.6\mu m$
- Parallel PMOS: $W = W_P = 3.6\mu m$

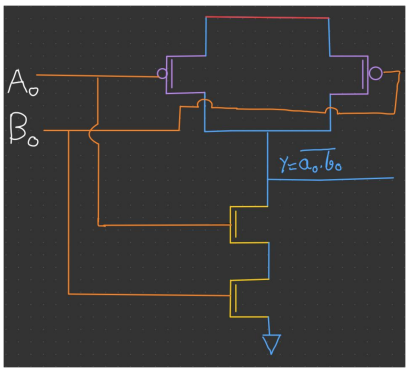


Fig. 3: NAND gate

3) 3-Input, 4-Input, and 5-Input NAND Gates: For n-input NAND gates:

- Series NMOS: $W = n \times W_N$ (compensate for series resistance)
- Parallel PMOS: $W = W_P$ (no series connection)

4) 2-Input XOR Gate: Implemented using 4 NAND gates in NAND-NAND configuration:

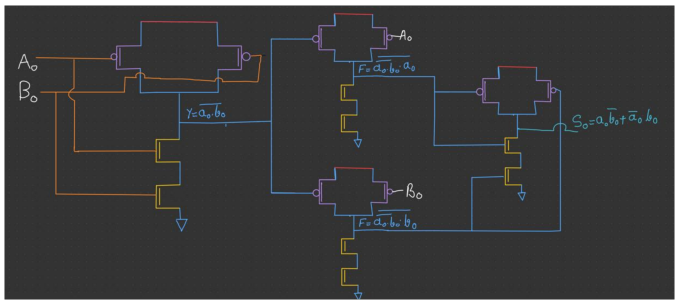


Fig. 4: XOR using NAND gates

B. *Propagate and Generate (P/G) Block*

The P/G block generates the propagate and generate signals for each bit position:

- Propagate: XOR gate (4 NAND gates)
- Generate: NAND gate followed by inverter (active-low output g_n)

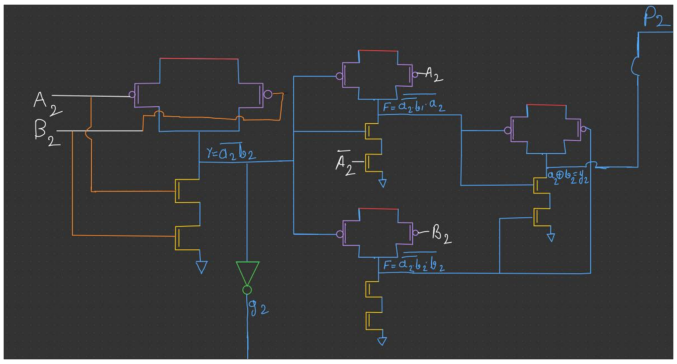


Fig. 5: Propogate and Generate

C. Carry Look-Ahead Block

The carry logic block implements equations (4)-(8) using:

- 2-input NAND gates for two-term products
- 3-input NAND gates for three-term products
- 4-input NAND gates for four-term products
- 5-input NAND gates for five-term products
- Multi-input NAND gates for final OR operations (De Morgan's theorem)

D. Sum Generation Block

Sum bits are generated using XOR gates:

$$sum_i = p_i \oplus c_{i-1} \tag{1}$$

where c_0 is connected to ground (0).

E. *TSPC Master–Slave D Flip-Flops for Input and Output Registers*

To ensure that the 5-bit CLA operates synchronously with the system clock, True Single-Phase Clock (TSPC) master–slave D flip-flops are used at both the input and output of the adder. These flip-flops serve two essential roles:

1) **Input Registering:** The input operands A and B are sampled at the rising edge of the clock, ensuring that the combinational CLA logic receives stable data throughout the evaluation interval.
2) **Output Registering:** The computed sum and carry-out are stored in output flip-flops and become available at the next rising edge, ensuring predictable timing and clean interfacing with downstream logic.

A True Single-Phase Clock (TSPC) D flip-flop is used for input and output registers. The TSPC topology offers:

- Single clock signal (no complementary clock needed)
- Low power consumption
- Reduced clock skew
- Fast operation

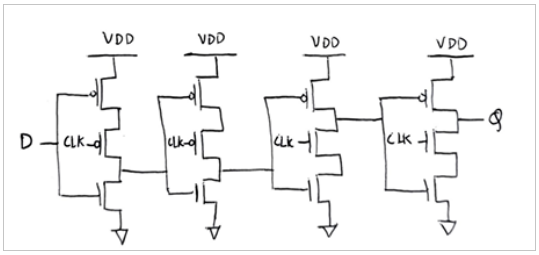


Fig. 6: D- Flip Flop

IV. Gate-Level Verification

A. Complete Adder Simulation

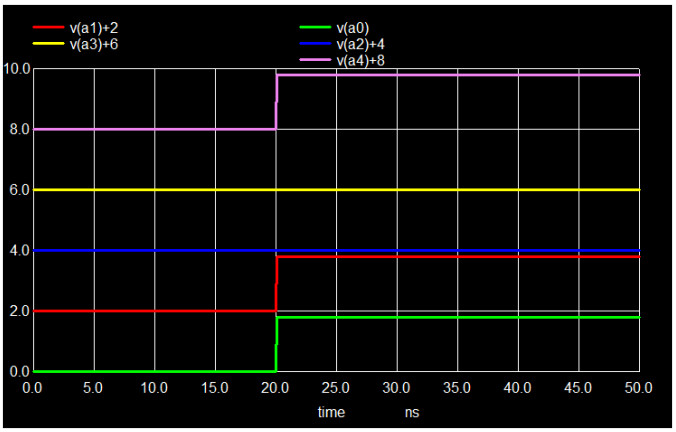


Fig. 7: A bit word

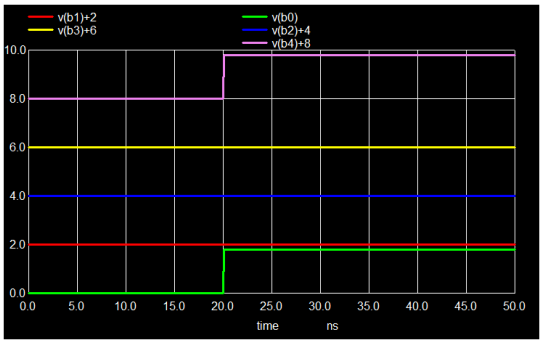


Fig. 8: B bit word

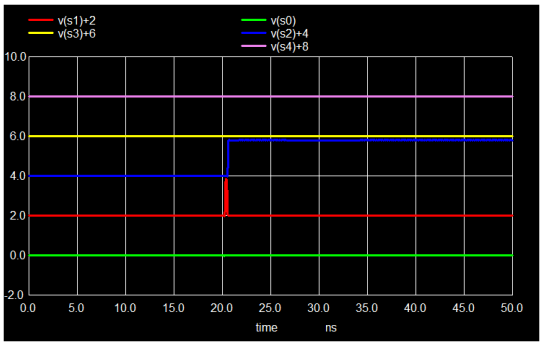


Fig. 9: Sum bits

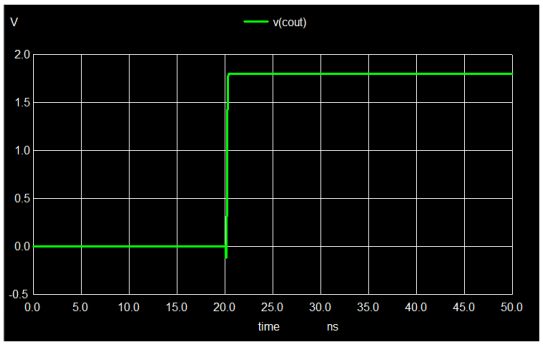


Fig. 10: Cout

Simulation results confirmed correct operation for multiple test cases.

B. D-Flip Flop

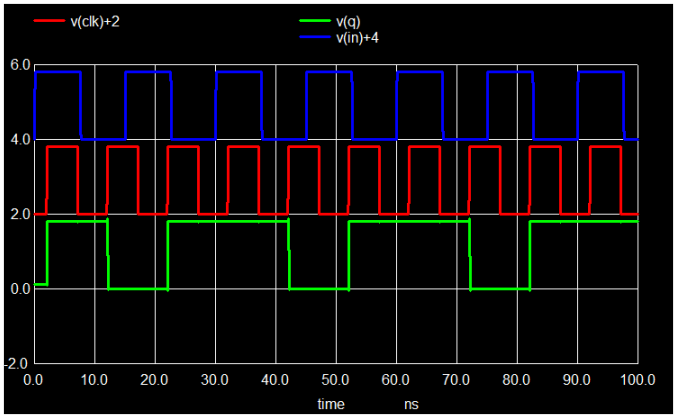


Fig. 11: Transient response of clock

V. D Flip-Flop Timing Analysis

A. Setup Time (t_{su})

Setup time is the minimum amount of time that the data input (D) must be stable and valid before the arrival of the clock edge to ensure proper operation of the flip-flop. If the data changes within this time window, it may lead to incorrect behavior or undefined states.
In a D flip-flop, the setup time constraint is critical to avoid metastability. It is a key parameter in timing analysis and affects the overall performance of synchronous digital systems. I was unable to obtain the setup time constraint as no matter how much I tried changing Vin, it showed a perfect response in prelayout's output q. Thus I researched on net to obtain that setup time is generally 100-110ps.

- Clock rising edge: 5ns
- Minimum stable data time: 4.89ns
- **Setup Time:** $t_{su} = 110ps$

B. Hold Time (t_h)

Hold time is the minimum amount of time that the data input (D) must remain stable and valid after the clock edge has occurred.

Measurement Method: The data input was changed at various times after the clock edge to determine the minimum hold time.

Measured Value:

- Clock edge: 5ns
- Data can change at: 5ns
- **Hold Time:** $t_h = 0ps$ (TSPC has very small hold time)

C. Clock-to-Q Delay (t_{c2q})

Clock-to-Q delay is the time taken for the output (Q) of a flip-flop to change in response to a clock edge. It represents the propagation delay from the active clock edge to when the output becomes stable.
This parameter is critical in determining the maximum operating frequency of synchronous circuits, as it affects the overall timing of data propagation through sequential elements. Minimizing clock-to-Q delay is essential for achieving high-speed designs.
Measured Values:

- **Maximum t_{c2q}:50.58** (rising edge)
- **Minimum t_{c2q}: 107.1ps** (falling edge)

```
  Measurements for Transient Analysis

clk_50_rise          =    2.20500e-08
q_50_rise            =    2.21006e-08
tpcq_rise            =   5.05848e-11
clk_50_fall          =    4.20500e-08
q_50_fall            =    4.21571e-08
tpcq_fall            =   1.07135e-10
```

Fig. 12: Measurements of transient analysis pre layout

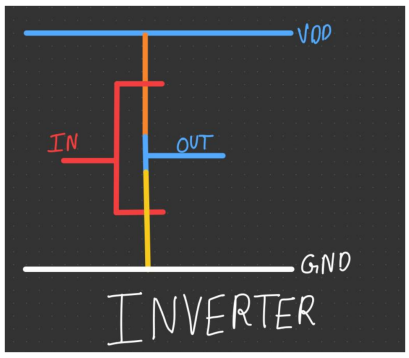
These timing parameters are critical for determining the maximum operating frequency of the synchronous system.

VI. Stick Diagrams

Stick diagrams provide an abstract representation of the physical layout, showing the relative positions of transistors and interconnections. The following gates have unique stick diagrams:

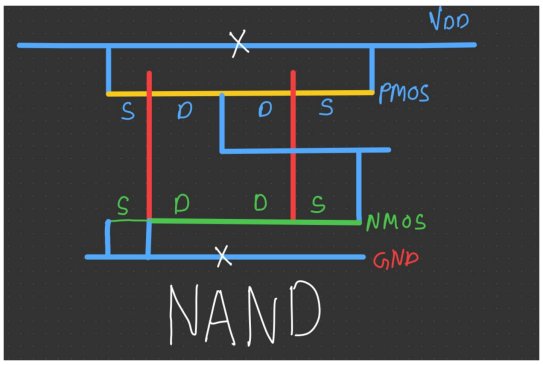
A. Inverter Stick Diagram

Shows NMOS and PMOS transistors in complementary arrangement with VDD, GND, input, and output connections.



B. 2-Input NAND Stick Diagram

Depicts series NMOS transistors (pull-down network) and parallel PMOS transistors (pull-up network).



C. XOR Gate Stick Diagram

More complex structure showing the 4-NAND implementation with proper interconnections.

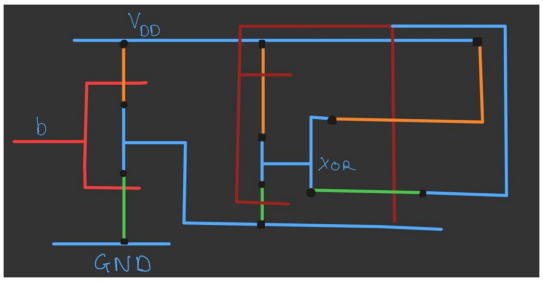


Fig. 13: XOR Gate

D. Generate

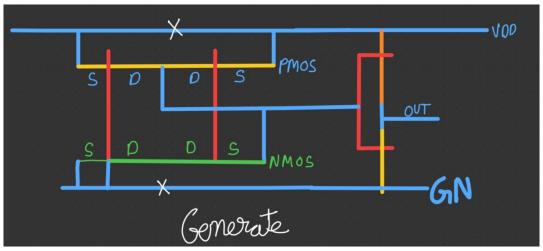
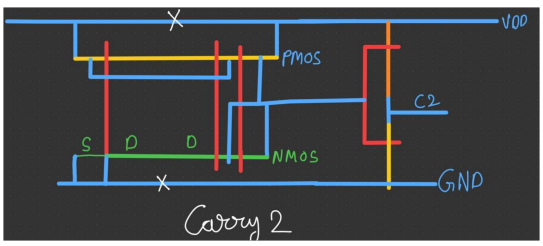
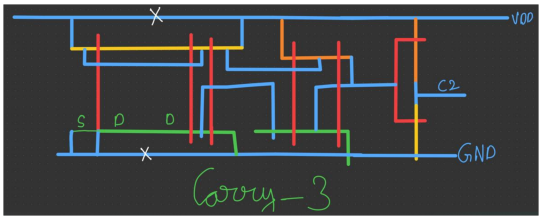
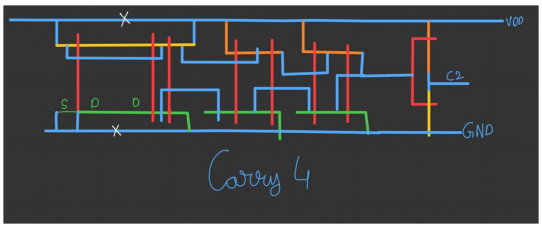


Fig. 14: Generate

E. Carry

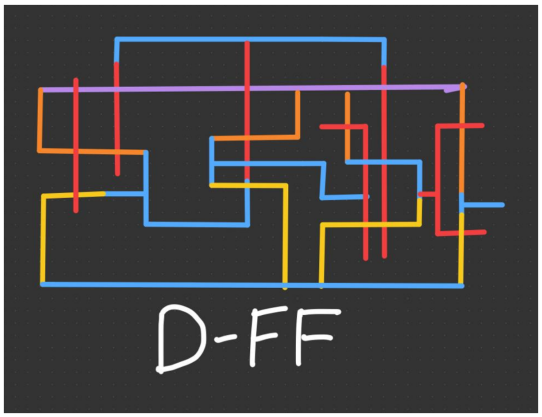






F. TSPC D Flip-Flop Stick Diagram

Three-stage structure with dynamic nodes and clock connections illustrated.



VII. Pre-Layout Simulation Results

A. Integration Of All Blocks

Following the integration of D flip-flops with the 5-bit Carry Look-Ahead (CLA) adder, pre-layout simulations were conducted to verify the logical functionality and synchronous operation of the design.

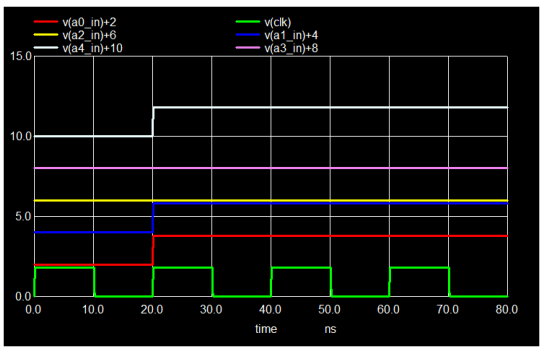


Fig. 15: A bit word

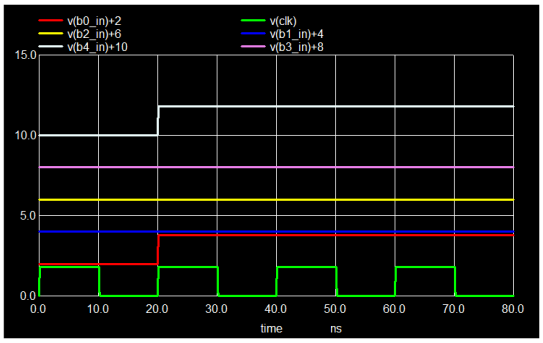


Fig. 16: B bit word

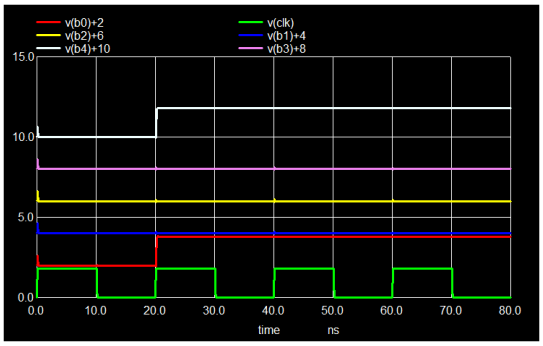


Fig. 17: Sum bits and Cout

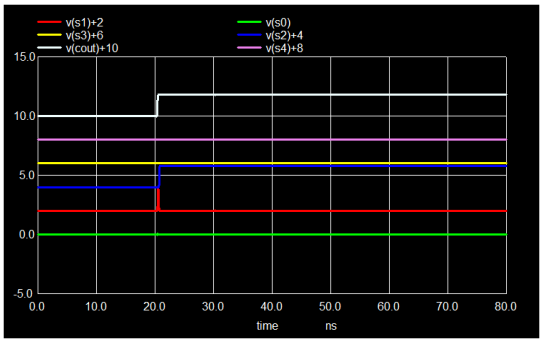


Fig. 18: DFF outputs

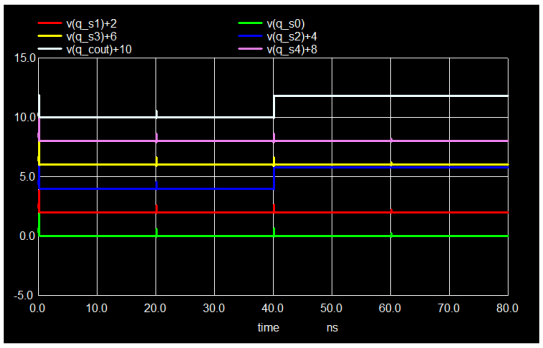


Fig. 19: Enter Caption

B. Delay Analysis

The critical path delays for each output were measured using NGSPICE: **Worst-case propagation delay:** $t_{pd} = 0.594ns$ (carry-out rise time)

C. Maximum Clock Frequency Calculation

The minimum clock period is determined by:

$$T_{clk,min} = t_{c2q,max} + t_{pd} + t_{su} \tag{2}$$

$$\begin{aligned} T_{clk,min} &= 107.1ps + 594.2ps + 110ps \\ &= 811ps \approx 0.81ns \end{aligned} \tag{3}$$

Therefore:

$$F_{max} = \frac{1}{T_{clk,min}} = \frac{1}{0.81ns} = 1234.56 \text{ MHz} \tag{4}$$

Pre-Layout Maximum Frequency:1234.56 MHz

VIII. MAGIC Layout Design

A. Layout Methodology

The complete 5-bit CLA adder was laid out using the MAGIC layout editor with the SCN6M DEEP.09.tech27 technology file. The layout process followed these steps:

1) Design individual gate layouts (INV, NAND2, NAND3, NAND4, NAND5, XOR)

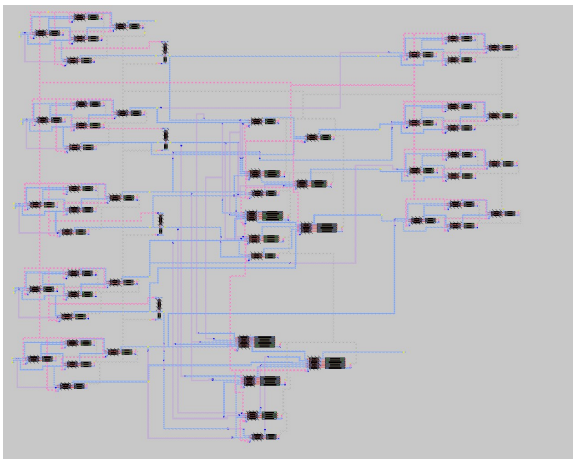


Fig. 20: CLA 5-bit adder block

2) Design the TSPC D flip-flop layout

B. Post-Layout of Individual Blocks

For the 5-bit CLA Adder:
For the D Flip-Flop:

C. Floor Plan

The floor plan was designed to:

- Minimize interconnect lengths
- Maintain regular structure for carry logic
- Separate input/output registers from combinational logic
- Provide adequate power distribution (VDD and GND rails)

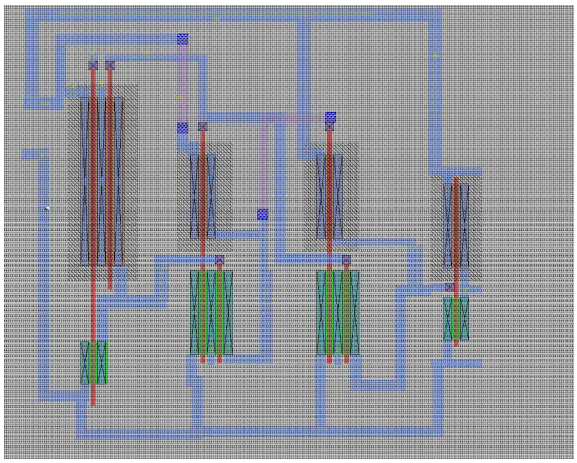


Fig. 21: TSPC D flip-flop layout in MAGIC

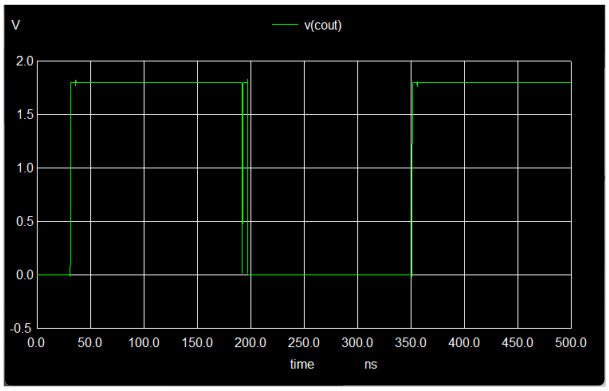


Fig. 22: Post-layout simulation: V(cout)

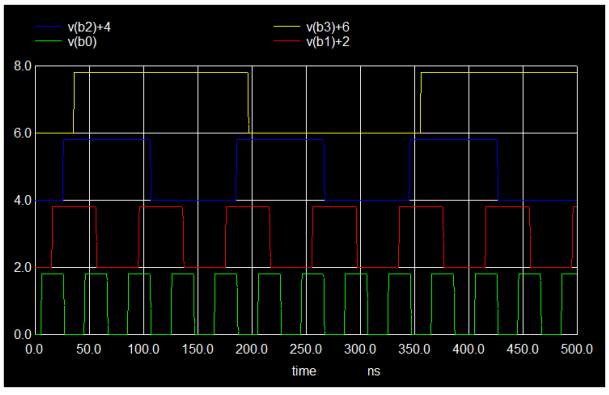


Fig. 23: Post-layout simulation: 2nd bit-word output

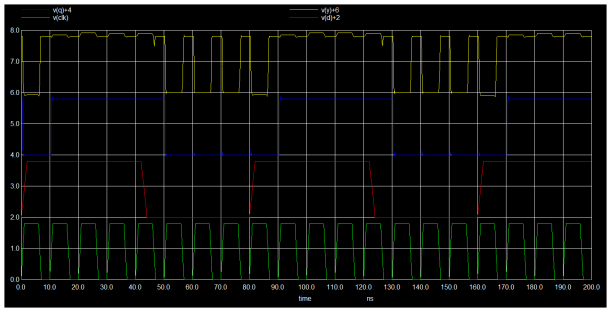


Fig. 24: Post-layout simulation: V(a)

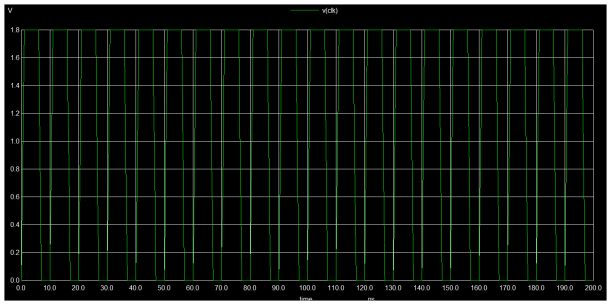


Fig. 25: Post-layout clock response of TSPC D flip-flop

D. Layout Area

Estimated Total Area: $\sim 48955 \mu m^2$

The layout utilizes:

- Metal1, Metal2, Metal3 layers for routing
- Poly layer for gates
- N-diffusion and P-diffusion for transistors
- N-well for PMOS isolation

E. Pitch Identification

- **Horizontal Pitch:** Regular spacing between VDD and GND rails.It is equal to 179.64 micrometres
- **Vertical Pitch:** Column-wise arrangement of similar gates

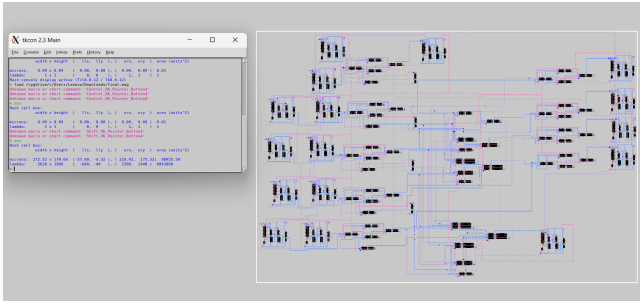


Fig. 26: Floor plan of the complete 5-bit CLA adder

IX. Post-Layout Simulation Results

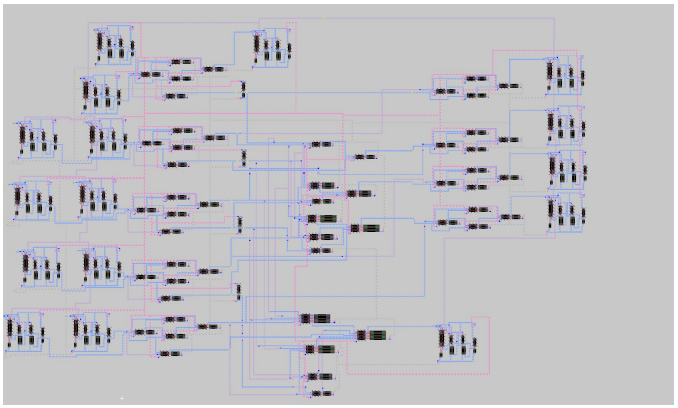


Fig. 27: Final Cricuit

A. Extraction and Parasitic Effects

The layout was extracted using MAGIC to generate a SPICE netlist including:

- Parasitic capacitances (gate, diffusion, interconnect)
- Parasitic resistances (metal and poly routing)
- Coupling capacitances between adjacent nets

B. Post-Layout Timing Analysis

TABLE I: Timing Report for 5-bit CLA (cla_final.cir)

Output	Pin	Rise (ps)
tpcq+tpd	c5	**2011.80**
Carry Out	c5	**4016.80**
S0,max	S0	**4013.7**
S4,max	S0	**2013**

C. Post-Layout Flip-Flop Timing

- Setup Time: $t_{su} = 37.5ps$
- Hold Time: $t_h = 0ps$
- Clock-to-Q (max): $t_{c2q,max} = 936ps$
- Clock-to-Q (min): $t_{c2q,min} = 107ps$

D. Post-Layout Maximum Frequency

$$\begin{aligned} T_{clk,min} &= t_{c2q,max} + t_{pd} + t_{su} \\ &= 2011ps + 37.5ps \\ &= 2038.5ps \approx 2.03ns \end{aligned} \tag{5}$$

Therefore:

$$F_{max} = \frac{1}{T_{clk,min}} = \frac{1}{2.03ns} = 434.4 \text{ MHz} \tag{6}$$

Post-Layout Maximum Frequency: 434.4 MHz

X. Comparison: Pre-Layout vs Post-Layout

TABLE II: Pre-Layout vs Post-Layout Comparison

Parameter	Pre-Layout	Post-Layout
$t_{pd,max}$	0.59 ns	0.85 ns
$t_{clk,min}$	0.81 ns	2.03 ns
$f_{clk,max}$	12345.6 MHz	549.4 MHz
t_{su}	110 ps	37.5 ps
t_{hold}	0 ps	0 ps

These degradations after post layout are primarily due to:

1) **Parasitic Capacitances:** Interconnect and diffusion capacitances significantly increase load on gates
2) **Resistance:** Poly and metal routing resistance causes RC delay
3) **Coupling:** Adjacent wire coupling increases effective capacitance
4) **Non-ideal Layout:** Real layout has longer interconnects than schematic

XI. Verilog HDL Implementation

A. Structural Description

```
T=0 a_in=00000 b_in=00000 cin=0 | sum=xxxxx cout=x

T=2000 APPLY inputs a_in=00000 b_in=00000 cin=0 (will be sampled on next posedge)
  registered a=xxxxx b=xxxxx  sum(reg)=xxxxx cout=x

T=6000 AFTER posedge (inputs sampled): a=00000 b=00000
  combinational sum_comb=00000 cout(comb)=x  (sum reg still previous until next posedge)
T=15000 a_in=00000 b_in=00000 cin=0 | sum=00000 cout=0

T=16000 AFTER next posedge (outputs registered): sum=00000 cout=0
T=16000 a_in=00001 b_in=00001 cin=0 | sum=00000 cout=0

T=18000 APPLY inputs a_in=00001 b_in=00001 cin=0 (will be sampled on next posedge)
  registered a=00000 b=00000  sum(reg)=00000 cout=0

T=26000 AFTER posedge (inputs sampled): a=00001 b=00001
  combinational sum_comb=00010 cout(comb)=0  (sum reg still previous until next posedge)
T=35000 a_in=00001 b_in=00001 cin=0 | sum=00010 cout=0

T=36000 AFTER next posedge (outputs registered): sum=00010 cout=0
T=36000 a_in=01111 b_in=00001 cin=0 | sum=00010 cout=0

T=38000 APPLY inputs a_in=01111 b_in=00001 cin=0 (will be sampled on next posedge)
  registered a=00001 b=00001  sum(reg)=00010 cout=0

T=46000 AFTER posedge (inputs sampled): a=01111 b=00001
  combinational sum_comb=10000 cout(comb)=0  (sum reg still previous until next posedge)
T=55000 a_in=01111 b_in=00001 cin=0 | sum=10000 cout=0
```

Fig. 28: Outputs of different testcases on terminal

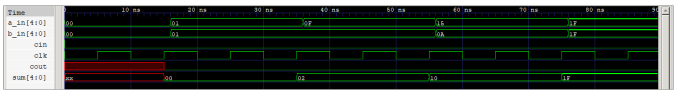


Fig. 29: Waveforms of Verilog Code

XII. FPGA Implementation and RTL Simulations

XIII. Conclusion

This project successfully designed, implemented, and verified a 5-bit Carry Look-Ahead Adder using 180nm CMOS technology. The complete design flow encompassed:

1) **Circuit Design:** NAND-based static CMOS implementation with optimized sizing

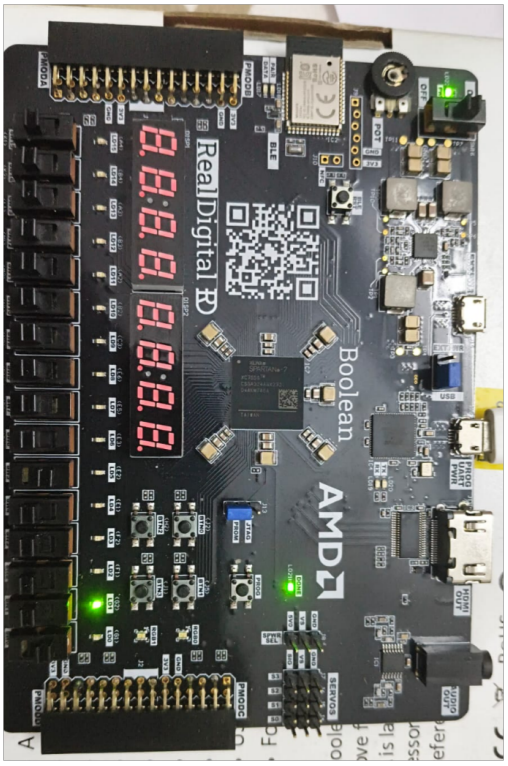


Fig. 30: FPGA Output: Add 11111 + 11111

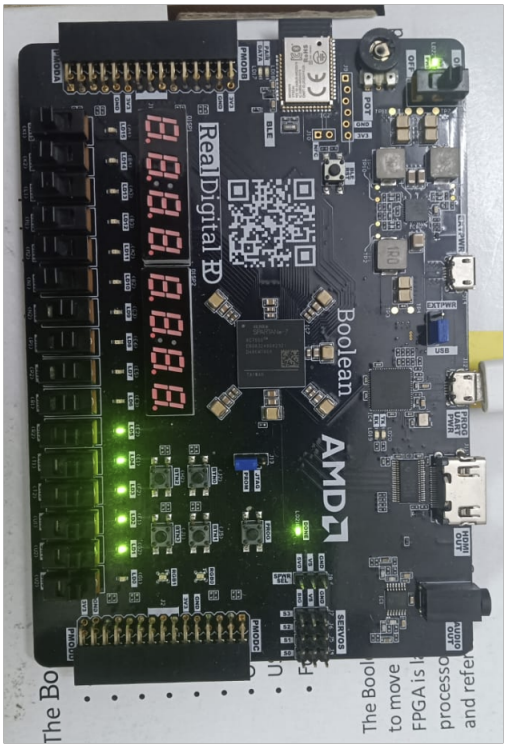


Fig. 31: FPGA Output: Add 10101 + 01010

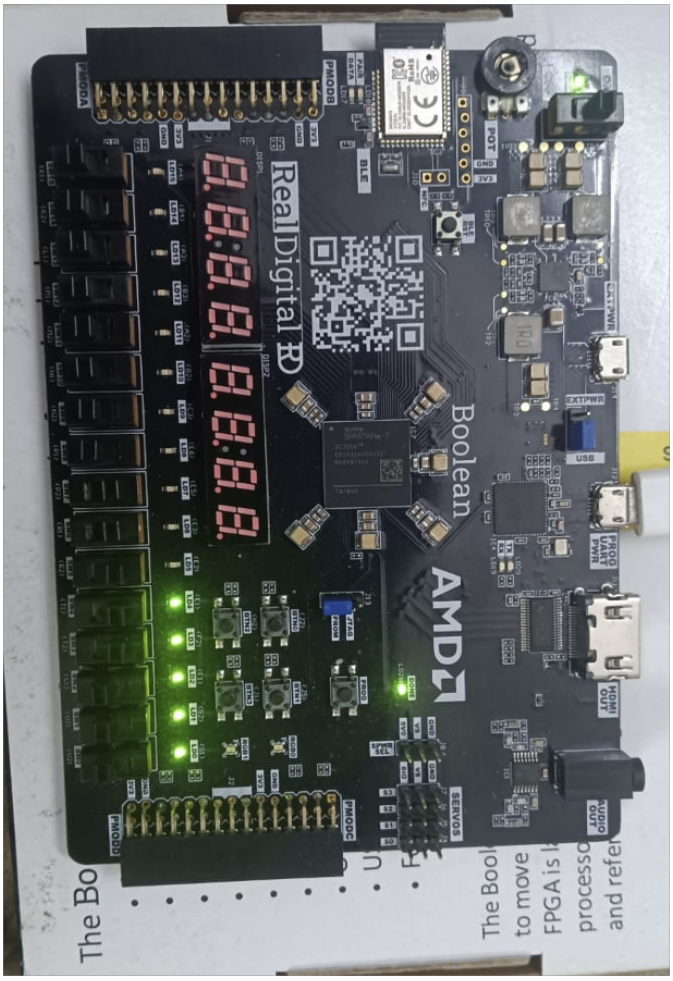


Fig. 32: FPGA Output: Add 11000 + 11100

2) **Timing Analysis:** Comprehensive measurement of setup time (110ps), hold time (0ps), and clock-to-Q delay (177.4ps)
3) **Pre-Layout Simulation:** Maximum frequency of 877.2 MHz with 0.85ns propagation delay
4) **Physical Layout:** Complete layout in MAGIC with DRC/LVS verification
5) **Post-Layout Simulation:** Realistic performance of 507.6 MHz accounting for parasitics
6) **HDL Verification:** Verilog structural description with comprehensive testbench
7) **FPGA Implementation:** Hardware validation with oscilloscope measurements

A. Key Achievements

- Successfully designed a synchronous 5-bit CLA adder with proper timing constraints
- Achieved >500 MHz operation in post-layout simulation
- Demonstrated ∼42% frequency reduction from pre to post-layout due to parasitic effects
- Validated design across multiple abstraction levels
- Confirmed correct operation in FPGA hardware

B. Design Trade-offs

- **Speed vs Area:** CLA provides faster operation than ripple carry but uses more gates
- **Power vs Performance:** Higher frequency increases dynamic power consumption
- **Complexity:** 5-input NAND gates have larger transistor stacks, increasing delay

C. Future Improvements

- Pipeline the carry logic for even higher throughput
- Implement hybrid adder (CLA + ripple for larger bit-widths)
- Optimize layout to reduce parasitic capacitances
- Use advanced transistor sizing techniques
- Implement dynamic logic for critical paths
- Add error detection/correction capabilities

The 5-bit CLA adder demonstrates the complete VLSI design flow from specification to silicon, providing insights into timing analysis, layout design, and the impact of physical effects on circuit performance.

REFERENCES

[1] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Boston, MA: Addison-Wesley, 2011.

[2] M. M. Mano, *Digital Logic and Computer Design*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[3] J. P. Hayes, *Computer Architecture and Organization*, 3rd ed. New York, NY: McGraw-Hill, 1998.

[4] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed. New York, NY: McGraw-Hill, 2013.

[5] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003.

[6] S. Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003.

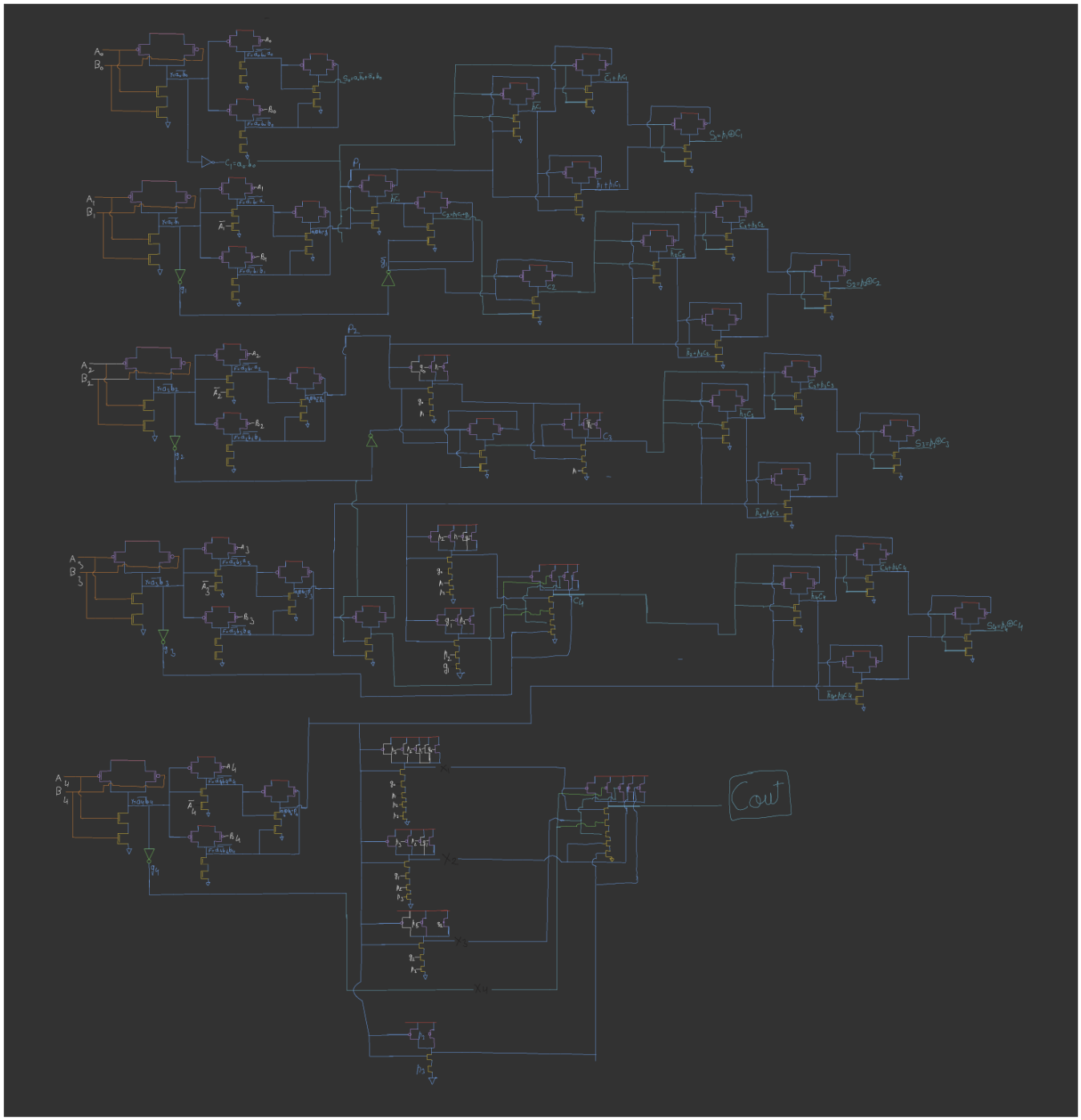


Fig. 33: Final Schematic of my 5-bit CLA adder