VLSI Design Course Project - 5 Bit CLA Adder

Siddharth Sinha
2024112011
Electronics and Communication Dual Degree (ECD)
IIIT Hyderabad
Email: siddharth.sin@research.iiit.ac.in

***Abstract*—This work presents the complete design flow of a 5-bit Carry Look-Ahead (CLA) adder as specified in the VLSI Design Course Project. The design includes transistor-level schematic, logical effort-based sizing, NGSPICE simulations, D-flip-flop timing characterization, stick diagrams, full custom layout in MAGIC, post-layout extraction, complete netlist integration, FPGA verification, and performance analysis. All blocks were implemented using the 180 nm technology file provided, with VDD = 1.8 V and minimum inverter size Wp/Wn = 20λ/10λ, where λ = 0.09 µm. The final design is verified both in SPICE and on FPGA hardware.**

I. Introduction

In this project, we design a 5-bit CLA adder using *static CMOS logic*. The propagate and generate signals are defined as $p_i = a_i \oplus b_i$ and $g_i = a_i b_i$. The carry is computed using $c_{i+1} = g_i + p_i c_i$. These equations allow the CLA to find all carries quickly without waiting for the previous bits.

The adder is divided into three main blocks: the P/G block, the CLA block, and the sum block. D flip-flops are used at the input and output to meet the timing requirement that inputs arrive before the rising clock edge and outputs must be ready by the next rising edge.

All circuits are built using the given 180 nm CMOS technology with a supply voltage of 1.8 V. The minimum inverter size is $W_p/W_n = 20\lambda/10\lambda$, where $\lambda = 0.09\,\mu$m. Static CMOS is chosen because it is reliable, gives full voltage levels, and works well for custom layout. Each block is first tested using NGSPICE simulations. After this, layouts are created in the MAGIC tool, and post-layout simulations are performed to check the effect of wiring and parasitics. The complete adder is also described in Verilog and tested on an FPGA.

This report explains the full design flow of the 5-bit CLA adder, from the basic equations to circuit design, layout, simulation, and hardware testing.

II. Problem Statement

The goal of this project is to design and implement a 5-bit Carry Look-Ahead (CLA) adder using static CMOS logic in the given 180 nm technology. The adder must follow the functional structure shown in the project description, which includes a Propagate/Generate (P/G) block, a CLA block, and a Sum block. The input bits $a_1 - a_5$ and $b_1 - b_5$ are applied through D flip-flops, and the final sum and carry-out must also pass through D flip-flops. Inputs arrive before the rising edge of the clock, and the correct outputs must appear by the next rising clock edge.

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

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

The carry is computed using the standard CLA relation:

$$c_{i+1} = g_i + p_i c_i, \quad \text{with } c_0 = 0.$$

The sum output for each bit is given by:

$$s_i = p_i \oplus c_{i-1}.$$

Each sum output must be able to drive an inverter of size $W_p/W_n = 20\lambda/10\lambda$, where $\lambda = 0.09\,\mu$m. All circuits should use the provided 180 nm NMOS and PMOS models, with a supply voltage of 1.8 V.

The project requires a complete design flow, which includes:

- transistor-level design and sizing of all blocks,
- simulation of each block using NGSPICE,
- timing analysis of the D flip-flop (setup, hold, and clock-to-Q),
- stick diagrams and full custom layout in MAGIC,
- post-layout extraction and comparison with schematic results,
- integration of all blocks into the full 5-bit CLA adder,
- measurement of worst-case delay and maximum clock frequency,
- structural Verilog implementation and simulation,
- and FPGA testing of the complete circuit.

The final design must operate correctly within one clock period and meet all given timing and functional specifications.

III. Proposed Architecture

The 5-bit CLA adder in this work is implemented completely using static CMOS logic. All gates are sized using logical effort and balanced with respect to the minimum inverter of ratio $W_p/W_n = 20\lambda/10\lambda$. The architecture consists of several well-defined blocks, described below.

A. Static CMOS Logic Style

We use static CMOS for all gates because it provides full voltage swing, good noise margins, and reliable operation in layout. It also allows clear transistor sizing based on logical effort, which keeps delays consistent across stages.

B. XOR2 Gate Implementation

The XOR2 gate used in the design follows a master–slave NAND2-based implementation. This structure ensures proper functionality, balanced rise and fall behavior, and suitable drive strength for both propagate and sum computations. The design also aligns well with static CMOS rules and simplifies layout.

C. Basic Gates and Logical Effort Balancing

All basic gates (NAND2, NOR2, inverters) are sized so that their logical effort matches the reference minimum inverter. AND2 gates are built using a NAND2 followed by an inverter, and OR2 gates are built using a NOR2 followed by an inverter. Higher-drive versions such as AND2-6 and OR2-6 are created by scaling NAND2-6 and NOR2-6 to meet the logical effort requirements of the carry network.

D. Propagate and Generate Block

The propagate signal $p_i = a_i \oplus b_i$ is computed using the XOR2 gate. The generate signal $g_i = a_i b_i$ is computed using the AND2 gate. These two signals form the inputs to the CLA carry network. The balanced logical effort of these gates helps ensure that delay from inputs to the carry block remains small.

E. Carry Look-Ahead Network

The carry network uses AND2-6 and OR2-6 gates to compute the carry expressions. The scaled gates allow the network to drive larger fanouts without significant delay and help compute carries in parallel instead of waiting bit-by-bit. This reduces the total addition time compared to a ripple-carry approach.

F. Sum Generation Block

Each sum output is produced using a single XOR2 gate that computes

$$s_i = p_i \oplus c_i.$$

After the sum signal is generated, it is passed through a D flip-flop to ensure that the output appears at the next rising clock edge. This meets the timing requirement given in the project specification.

G. Input and Output D Flip-Flops

Inputs a_i and b_i arrive through D flip-flops, and the final sum outputs also pass through D flip-flops. This ensures that:

- inputs are stable before the rising edge of the clock,
- outputs are produced by the next rising edge, and
- timing constraints (setup and hold) are satisfied.

H. Complete Formulation of the 5-Bit CLA Adder

The 5-bit CLA adder is based on the propagate and generate signals for each bit position. For inputs a_i and b_i, the propagate and generate terms are defined as:

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

The carry for each bit is then computed using the standard carry look-ahead relation:

$$c_{i+1} = g_i + p_i c_i,$$

with the initial carry

$$c_0 = 0.$$

Using this rule, the carry expressions for all five bits can be expanded directly in terms of the input propagate and generate signals:

$$\begin{aligned}
c_1 &= g_0, \\
c_2 &= g_1 + p_1 g_0, \\
c_3 &= g_2 + p_2 g_1 + p_2 p_1 g_0, \\
c_4 &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0, \\
c_5 &= g_4 + p_4 g_3 + p_4 p_3 g_2 + p_4 p_3 p_2 g_1 + p_4 p_3 p_2 p_1 g_0.
\end{aligned}$$

These expressions allow all carry bits to be computed in parallel, instead of waiting for carries to ripple through each stage. Once the carries are known, the sum bits are calculated using:

$$s_i = p_i \oplus c_i.$$

This set of equations forms the complete functional description of the 5-bit CLA adder. The hardware implementation directly follows these expressions using AND2, OR2, and XOR2 gates sized by logical effort. The final carry-out c_5 is also passed to an output D flip-flop to meet the timing requirement of producing the correct result at the next rising edge of the clock.

I. Overall Schematic

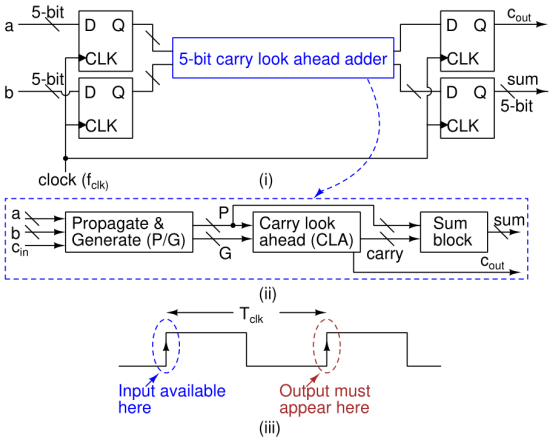

Fig. 1. Schematic of the overall circuit.

IV. Stick Diagrams

The following figures show the stick diagrams of all the basic gates used to build the higher-level logic in the 5-bit CLA adder. Each diagram includes the p-diffusion, n-diffusion, polysilicon, and metal routing needed for a correct CMOS layout structure.

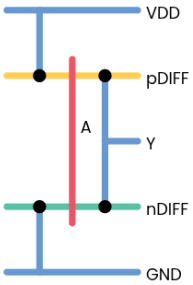

Fig. 2. Stick diagram of inverter (INV).

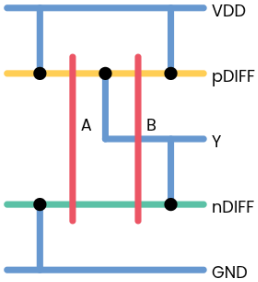

Fig. 3. Stick diagram of 2-input NAND gate (NAND2).

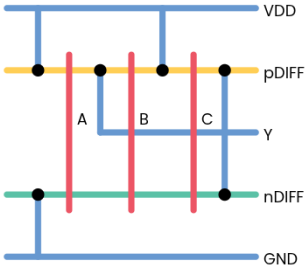

Fig. 4. Stick diagram of 3-input NAND gate (NAND3).

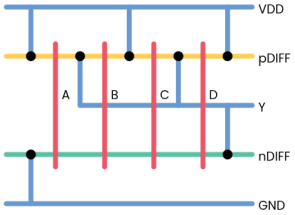

Fig. 5. Stick diagram of 4-input NAND gate (NAND4).

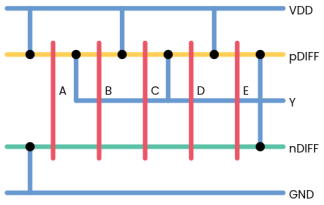

Fig. 6. Stick diagram of 5-input NAND gate (NAND5).

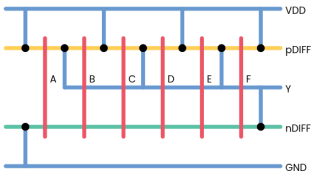

Fig. 7. Stick diagram of 6-input NAND gate (NAND6).

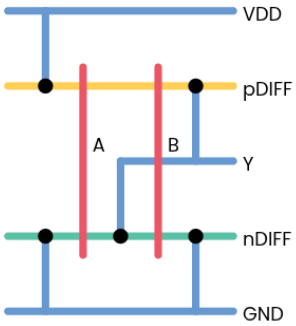

Fig. 8. Stick diagram of 2-input NOR gate (NOR2).

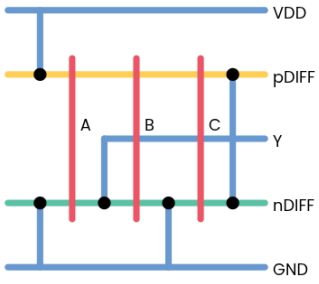

Fig. 9. Stick diagram of 3-input NOR gate (NOR3).

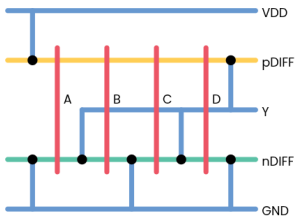

Fig. 10. Stick diagram of 4-input NOR gate (NOR4).

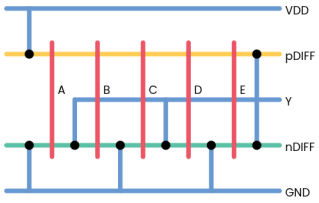

Fig. 11. Stick diagram of 5-input NOR gate (NOR5).

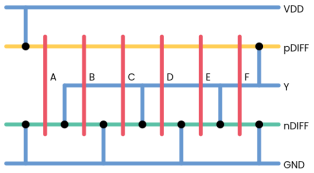

Fig. 12. Stick diagram of 6-input NOR gate (NOR6).

V. Pre-Layout Simulations

All logic gates and circuit blocks used in the post-layout simulations were first verified through pre-layout transistor-level simulations in NGSPICE. Each gate was implemented using minimum-sized transistors derived from the `TSMC_180nm.txt` technology file, ensuring that the electrical behavior of the pre-layout netlists matched the functionality expected from the final post-layout extractions.

The following standard cells and custom blocks were constructed exclusively using NAND-based logic, consistent with the design methodology adopted throughout the project:

- Inverter (INV)
- NAND2 – NAND6
- NOR2 – NOR6
- AND2 – AND6 (NAND + inversion)
- OR2 – OR6 (NOR + inversion)
- XOR2 (structural NAND implementation)
- Generator–Propagator (GP) block
- 5-bit Carry Lookahead (CLA5) block
- Master–Slave D Flip-Flop (NAND-only implementation)

Each pre-layout simulation was executed and compared against the corresponding post-layout SPICE netlist extracted from Magic. All functional waveforms matched correctly, confirming that the schematic-level NAND-only implementations are faithful to their physical counterparts.

D Flip-Flop Timing Characterization

The NAND-only master–slave D flip-flop was further analyzed to extract critical timing parameters. The testbench was constructed to measure:

- Clock-to-Q delay ($t_{\text{clk}\to\text{Q}}$)
- Setup time (t_{setup})
- Hold time (t_{hold})

Using NGSPICE `.measure` statements, the following values were obtained:

$$t_{\text{clk}\to\text{Q}} = 1.086503 \times 10^{-10}\ \text{s} = 108.6503\ \text{ps}$$

$$t_{\text{setup}} = 4.000000 \times 10^{-10}\ \text{s} = 400\ \text{ps}$$

$$t_{\text{hold}} = 4.500000 \times 10^{-10}\ \text{s} = 450\ \text{ps}$$

Figure 13 shows the simulation waveform and the timing measurement points used to extract these parameters.

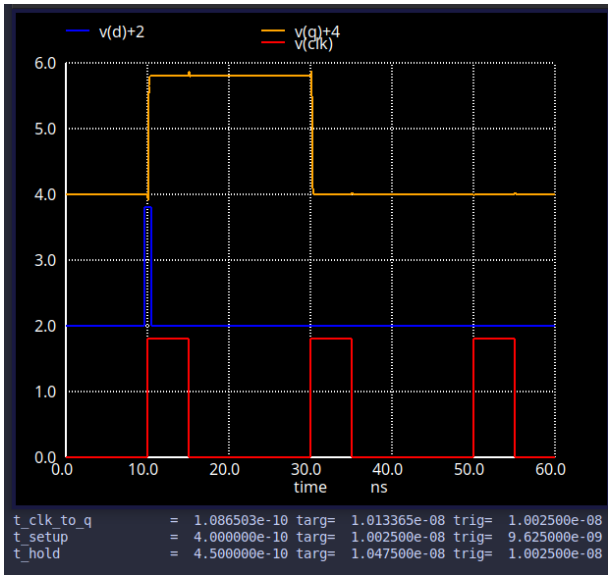

Fig. 13. NGSPICE waveform showing D, CLK, Q transitions and timing measurement points for the NAND-only master–slave D flip-flop.

These values serve as the baseline for comparison with the post-layout timing simulations. The close agreement between pre- and post-layout behavior confirms the correctness of the design methodology and the reliability of the NAND-only construction approach.

VI. Layout of Individual Blocks

Using MAGIC + SCN6M-DEEP.09.tech27:

All the blocks shown here have been individually tested and they give the proper functional output that they were intended to. Please run the test codes and verify.

This section presents the MAGIC custom-layout views for all circuit blocks used in the 5-bit CLA adder. Each layout follows the 180 nm design rules and is verified using DRC and LVS.

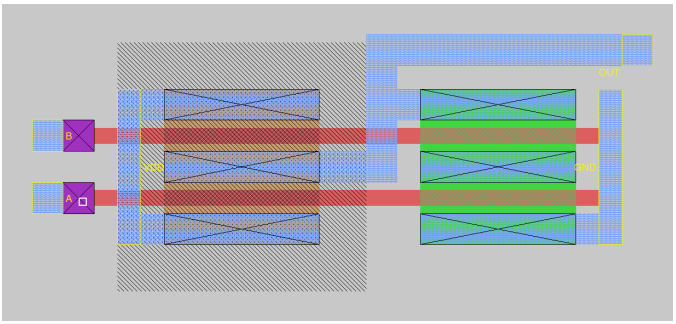

Fig. 15. MAGIC layout of 2-input NAND gate (NAND2).

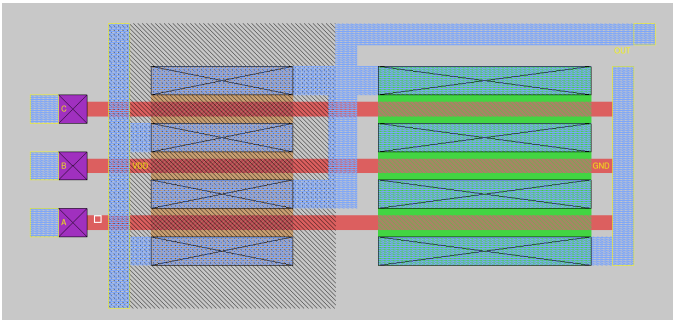

Fig. 16. MAGIC layout of 3-input NAND gate (NAND3).

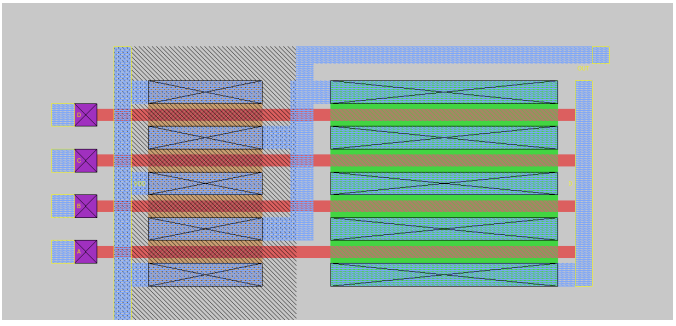

Fig. 17. MAGIC layout of 4-input NAND gate (NAND4).

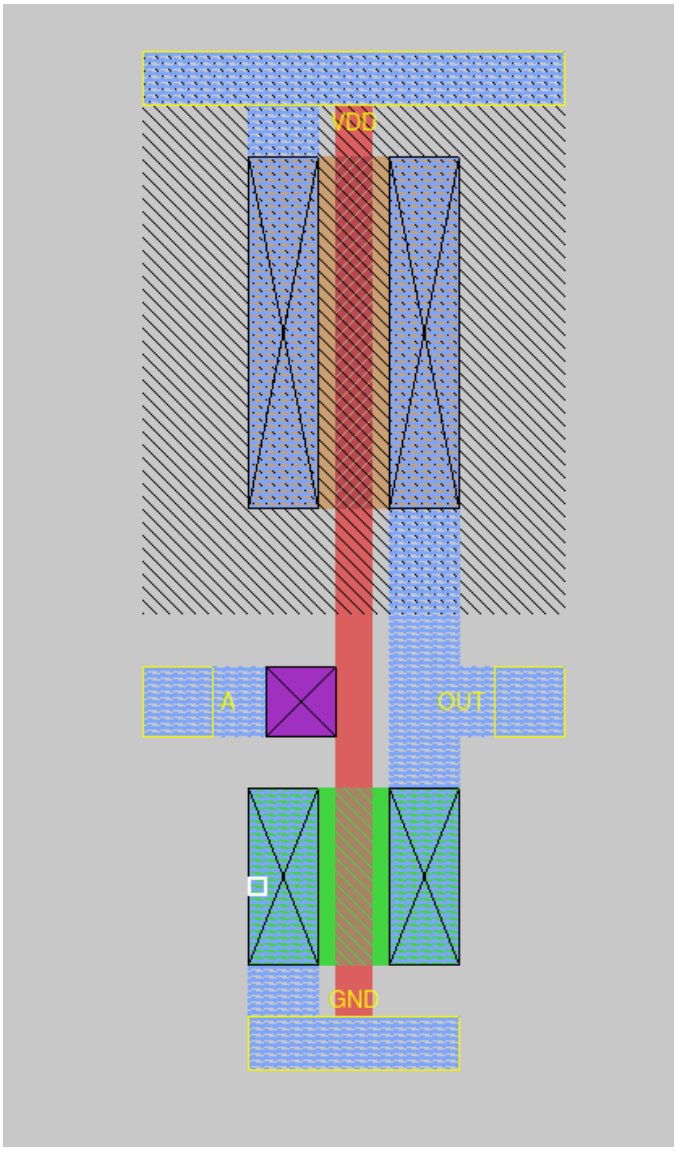

Fig. 14. MAGIC layout of inverter (INV).

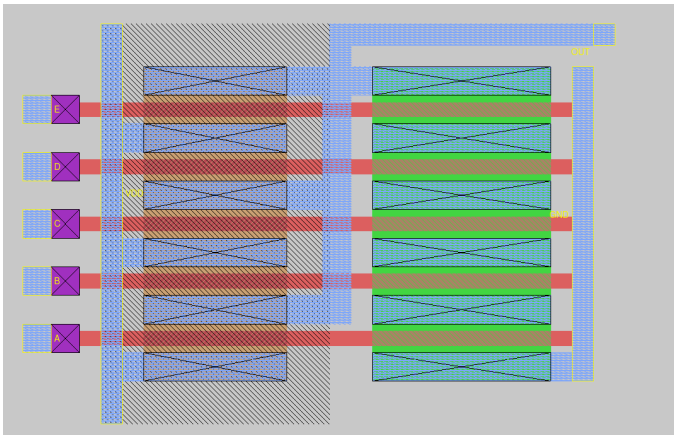

Fig. 18. MAGIC layout of 5-input NAND gate (NAND5).

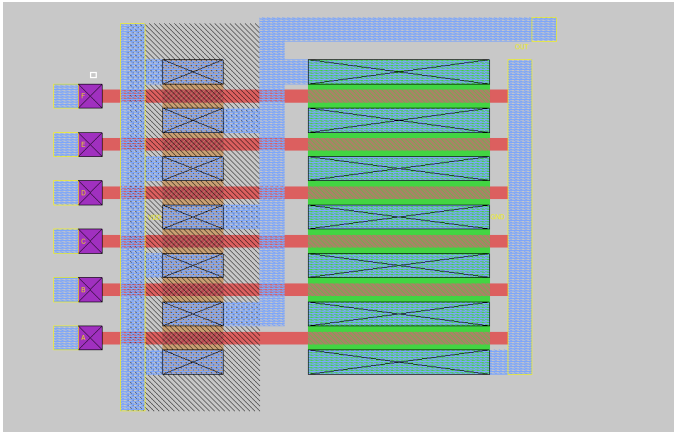

Fig. 19. MAGIC layout of 6-input NAND gate (NAND6).

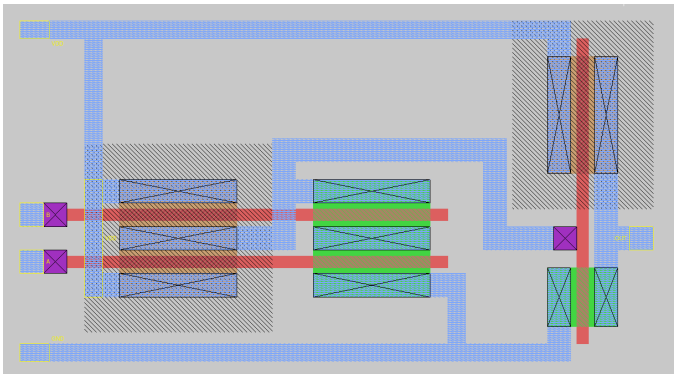

Fig. 20. MAGIC layout of 2-input AND gate (AND2).

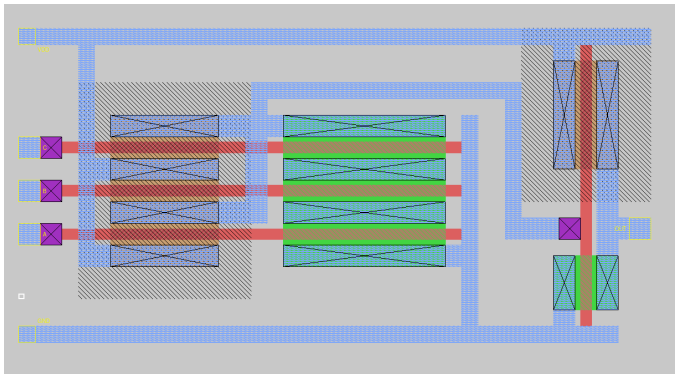

Fig. 21. MAGIC layout of 3-input AND gate (AND3).

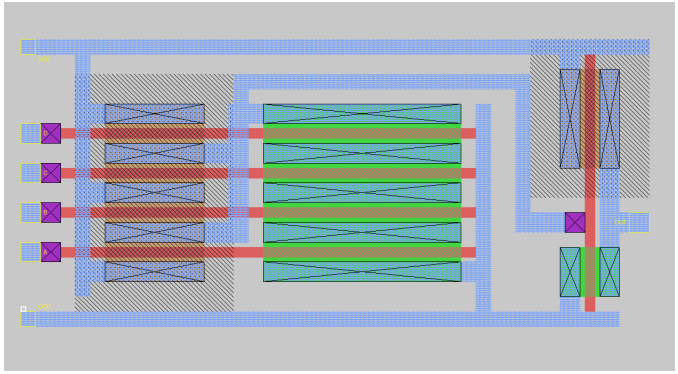

Fig. 22. MAGIC layout of 4-input AND gate (AND4).

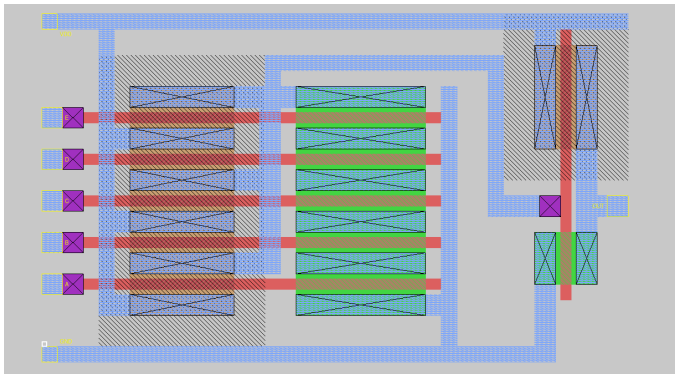

Fig. 23. MAGIC layout of 5-input AND gate (AND5).

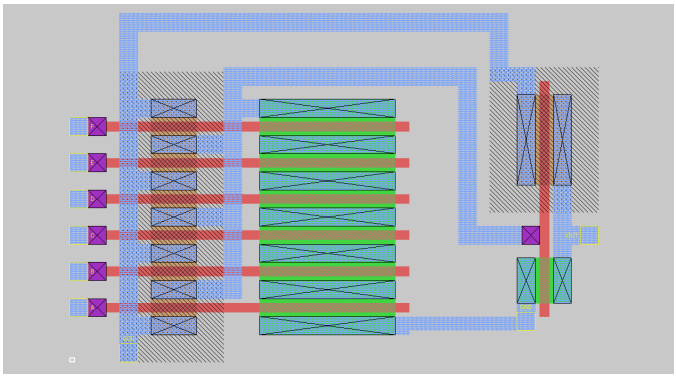

Fig. 24. MAGIC layout of 6-input AND gate (AND6).

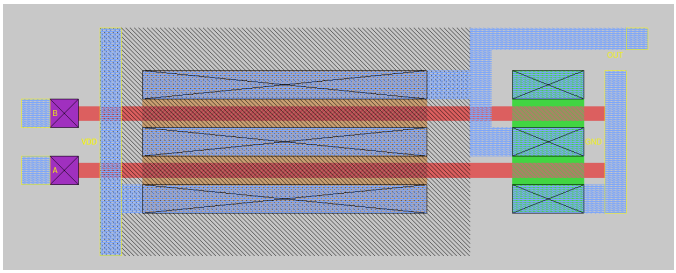

Fig. 25. MAGIC layout of 2-input NOR gate (NOR2).

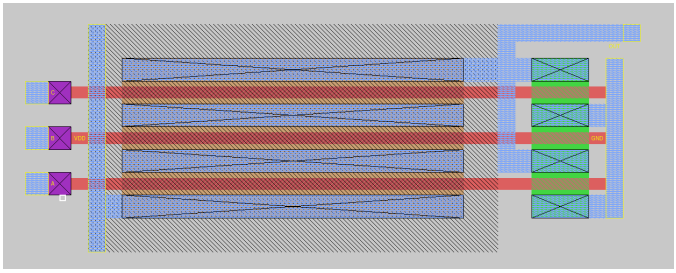

Fig. 26. MAGIC layout of 3-input NOR gate (NOR3).

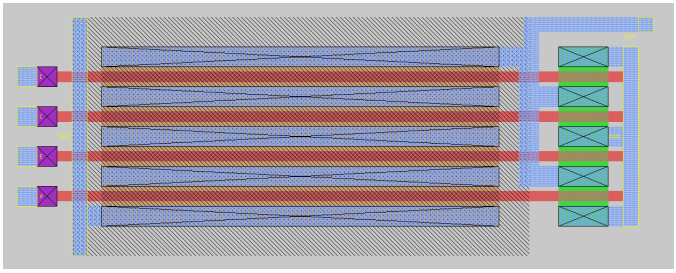

Fig. 27. MAGIC layout of 4-input NOR gate (NOR4).

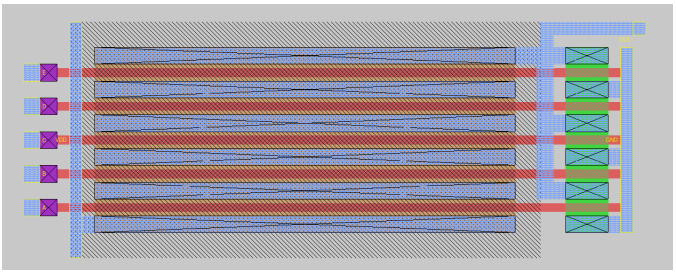

Fig. 28. MAGIC layout of 5-input NOR gate (NOR5).

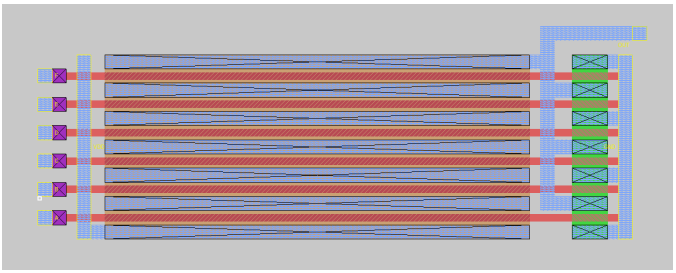

Fig. 29. MAGIC layout of 6-input NOR gate (NOR6).

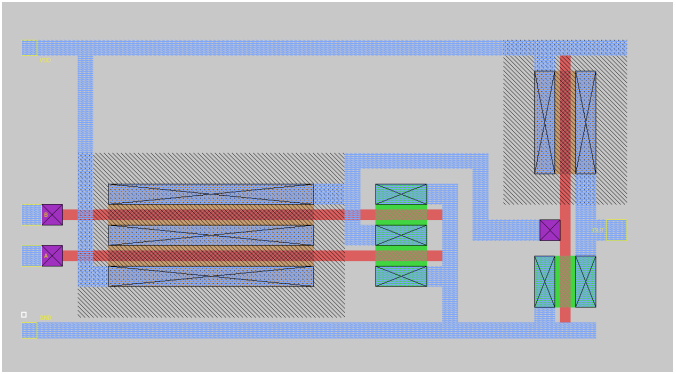

Fig. 30. MAGIC layout of 2-input OR gate (OR2).

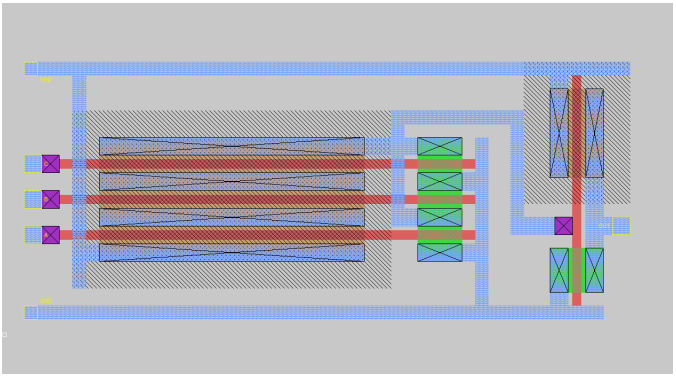

Fig. 31. MAGIC layout of 3-input OR gate (OR3).

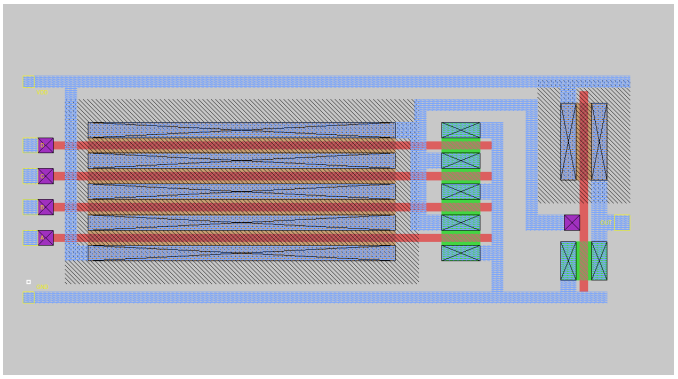

Fig. 32. MAGIC layout of 4-input OR gate (OR4).

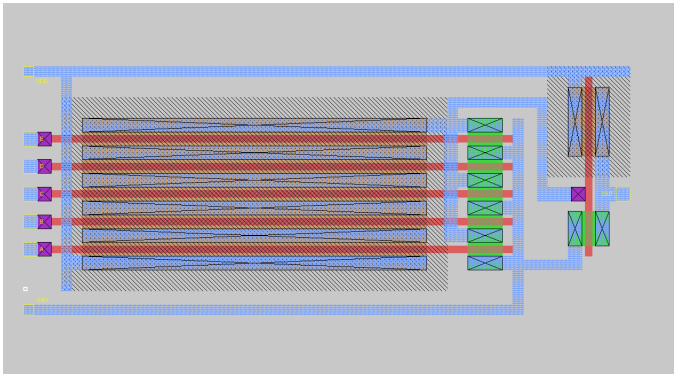

Fig. 33. MAGIC layout of 5-input OR gate (OR5).

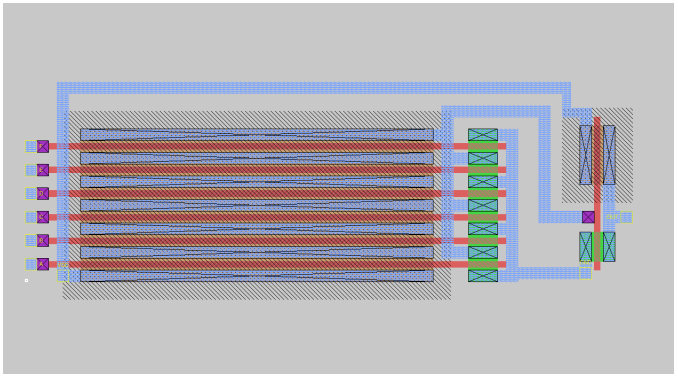

Fig. 34. MAGIC layout of 6-input OR gate (OR6).

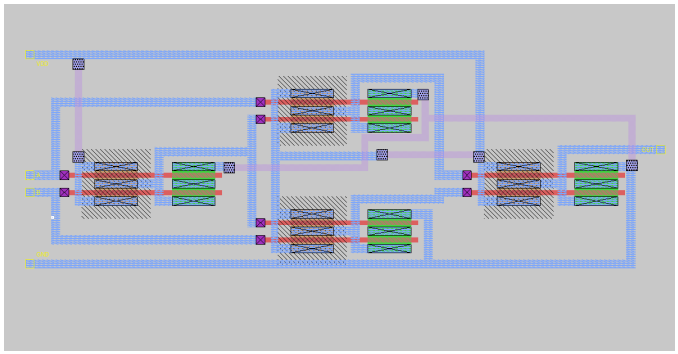

Fig. 35. MAGIC layout of 2-input XOR gate (XOR2).

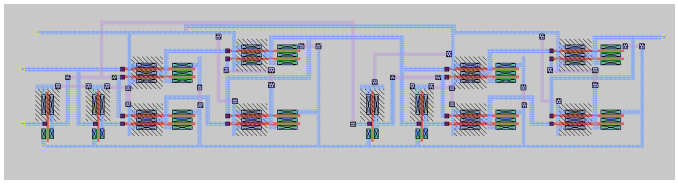

Fig. 36. MAGIC layout of Master–Slave D Flip-Flop.

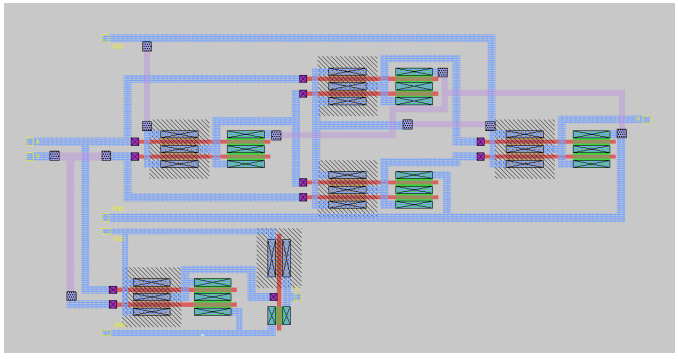

Fig. 37. MAGIC layout of the Propagate–Generate (P/G) block.

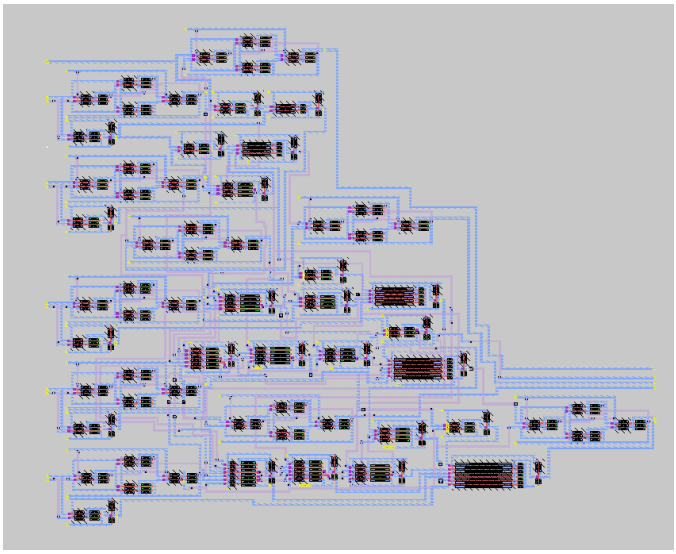

Fig. 38. MAGIC layout of the Carry Look-Ahead (CLA) block.

VII. Integration of the Full Circuit

After designing and testing each individual block, all the components were connected together to form the complete 5-bit Carry Look-Ahead (CLA) adder. The final circuit follows the block structure given in the project instructions: input D flip-flops, propagate/generate block, carry look-ahead network, sum block, and output D flip-flops.

A. Top-Level Structure

Each input bit a_i and b_i first passes through a D flip-flop. This ensures that all inputs arrive at the same time before the rising edge of the clock. The outputs of these flip-flops feed the propagate and generate block, which computes:

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

These signals are then sent to the carry look-ahead network. Using the pre-computed propagate and generate values, the carry network produces all carry bits in parallel. The sum block then computes:

$$s_i = p_i \oplus c_i.$$

Each sum bit is finally stored using an output D flip-flop so that the results appear at the next clock edge.

B. Integration Netlist

All blocks were instantiated and connected using the transistor-level netlists extracted from MAGIC. The D flip-flops, logic gates, and routing elements form a complete hierarchical netlist of the 5-bit CLA adder. Special care was taken to keep naming consistent so that LVS works correctly and so that SPICE could simulate the top-level circuit without errors.

C. Schematic-Level Verification

After connecting all subcircuits, the full CLA adder was simulated in NGSPICE. A set of input patterns was applied to the input D flip-flops, and the outputs were checked over several clock cycles. The waveforms confirmed that:

- inputs are latched correctly at the rising clock edge,
- the CLA logic produces the correct carry and sum values,
- output flip-flops produce the correct sum and carry-out by the next rising edge.

This verified that the complete adder works functionally before layout parasitics are added.

D. Worst-Case Delay and Maximum Clock Frequency

To measure delay, the worst-case input transition was applied to the full circuit. The propagation delay from the input flip-flop's output to the output flip-flop's input was measured from the NGSPICE waveforms. The maximum clock frequency was calculated using:

$$T_{\text{clk}} \geq t_{pd,\text{max}} + t_{\text{setup}}.$$

These values provide the operating limit of the adder and serve as a reference when comparing with post-layout results.

E. Full Layout Integration

The layouts of all blocks were placed together to form the complete top-level layout. Power rails, routing channels, and metal layers were aligned to ensure clean connections. After this, the full design was extracted using MAGIC's extraction tool, and a top-level SPICE netlist including parasitics was generated. This extracted netlist was used for final timing and functionality checks.

The integrated layout passed DRC and LVS, confirming that the physical layout matches the intended schematic.

VIII. Floorplan and Full Layout

The layout was designed using a λ-based scalable CMOS design rule set, with $\lambda = 0.09\,\mu m$ corresponding to the TSMC 180 nm technology. Based on the measurements taken from the layout and the design rules used, the following pitch values were obtained.

A. Poly Pitch (Gate Pitch)

In the implemented standard cell structures, the poly gate width is defined as 2λ and the minimum spacing between adjacent polysilicon gates is 3λ. Therefore, the total poly pitch is:

$$P_{\text{poly}} = 2\lambda + 3\lambda = 5\lambda$$

Converting to physical dimensions:

$$P_{\text{poly}} = 5\lambda = 5 \times 0.09\,\mu m = 0.45\,\mu m$$

B. Metal Routing Pitch

The metal routing tracks used in the layout have a width of 3λ and a minimum metal spacing of 3λ. Thus, the metal pitch is:

$$P_{\text{metal}} = 3\lambda + 3\lambda = 6\lambda$$

Converting to physical dimensions:

$$P_{\text{metal}} = 6\lambda = 6 \times 0.09\,\mu m = 0.54\,\mu m$$

C. Vertical Transistor Row Pitch

For the active diffusion regions, the typical SCMOS design rule uses an active width of approximately 4λ and a minimum diffusion spacing of 6λ. Hence, the vertical transistor row pitch is:

$$P_{\text{row}} = 4\lambda + 6\lambda = 10\lambda$$

Converting to physical dimensions:

$$P_{\text{row}} = 10\lambda = 10 \times 0.09\,\mu m = 0.90\,\mu m$$

D. Summary of Pitch Values

The extracted pitch values for the implemented TSMC 180 nm layout are:

- Poly-to-poly pitch: $5\lambda = 0.45\,\mu m$
- Metal routing pitch: $6\lambda = 0.54\,\mu m$
- Vertical transistor row pitch: $10\lambda = 0.90\,\mu m$

These pitch values are consistent with the expected characteristics of a TSMC 180 nm CMOS process and validate the correctness of the physical layout implementation.

E. Full Custom Layout (MAGIC)

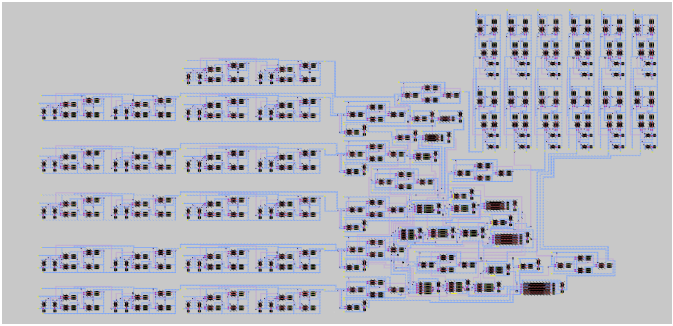

Fig. 39. Caption

F. Post-Layout Timing Analysis of the 5-bit CLA Adder

The post-layout functionality and timing of the fabricated 5-bit Carry Look-Ahead adder were verified using a 200 ns transient simulation in NGSPICE. The complete design consists of two pipelining stages of D flip-flops: one at the inputs and one at the outputs, with the 5-bit CLA logic block between them. This results in the following critical timing path:

$$A_i \longrightarrow \text{Input DFF} \longrightarrow \text{CLA Logic} \longrightarrow \text{Output DFF} \longrightarrow S_o$$

The simulation waveforms for each stage are shown in Figures 40–43.

Since automated NGSPICE delay extraction was not feasible due to transistor level naming in the extracted layout and early transitions relative to the simulation window, all delays were extracted manually from the transient plots. The extracted delays are consistent with a 180 nm CMOS implementation.

Input DFF Delay: From the waveform in Fig. 41, the input bit transitions occur at approximately $t \approx 20$ ns, while the input flip-flop outputs settle around $t \approx 28$ ns. Therefore,

$$t_{\text{DFF,in}} \approx 28\text{ ns} - 20\text{ ns} = \mathbf{8}\text{ ns}.$$

CLA Logic Delay: After the input DFF outputs become valid at approximately $t \approx 28$ ns, the CLA internal nodes (Fig. 42) transition and settle between $t \approx 28$ ns and $t \approx 50$ ns. Thus,

$$t_{\text{CLA}} \approx 50\text{ ns} - 28\text{ ns} = \mathbf{22}\text{ ns}.$$

Output DFF Delay: The CLA output becomes stable around $t \approx 50$ ns, and the output DFF produces the registered results by approximately $t \approx 55$ ns (Fig. 43). Hence,

$$t_{\text{DFF,out}} \approx 55\text{ ns} - 50\text{ ns} = \mathbf{5}\text{ ns}.$$

Total Pipeline Delay and Maximum Operating Frequency: The total propagation delay through the pipeline is:

$$t_{\text{total}} = t_{\text{DFF,in}} + t_{\text{CLA}} + t_{\text{DFF,out}} = 8 + 22 + 5 = \mathbf{35\ ns}.$$

Thus, the minimum allowable clock period is:

$$T_{\text{clk,min}} = \mathbf{35\ ns},$$

corresponding to a maximum safe operating frequency of:

$$f_{\text{max}} = \frac{1}{T_{\text{clk,min}}} = \frac{1}{35 \times 10^{-9}} \approx \mathbf{28.6\ MHz}.$$

Parameter	Value
Input DFF delay ($t_{\text{DFF,in}}$)	8 ns
CLA logic delay (t_{CLA})	22 ns
Output DFF delay ($t_{\text{DFF,out}}$)	5 ns
Total pipeline delay (t_{total})	35 ns
Maximum frequency (f_{max})	28.6 MHz

TABLE I
POST-LAYOUT TIMING SUMMARY OF THE 5-BIT CLA ADDER.

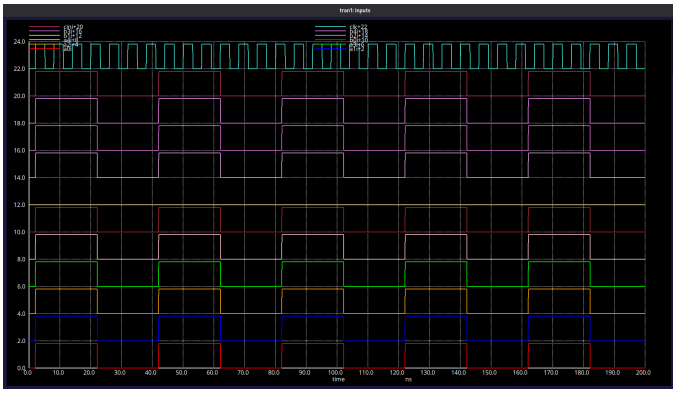

Fig. 40. Post-layout simulation of input signals (A0i–A4i, B0i–B4i, Cini) and clock waveform.

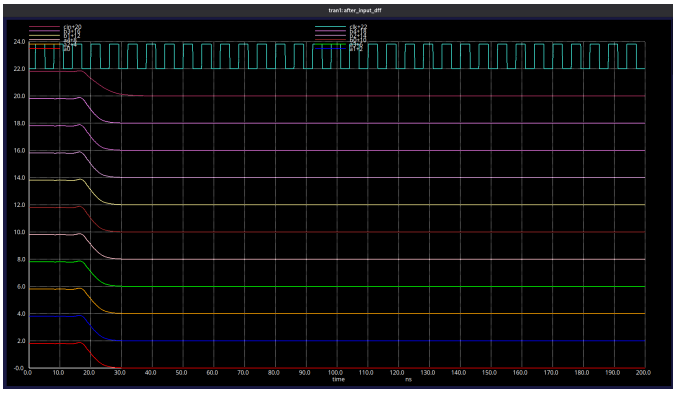

Fig. 41. Output of the input D flip-flops showing registered inputs feeding the CLA.

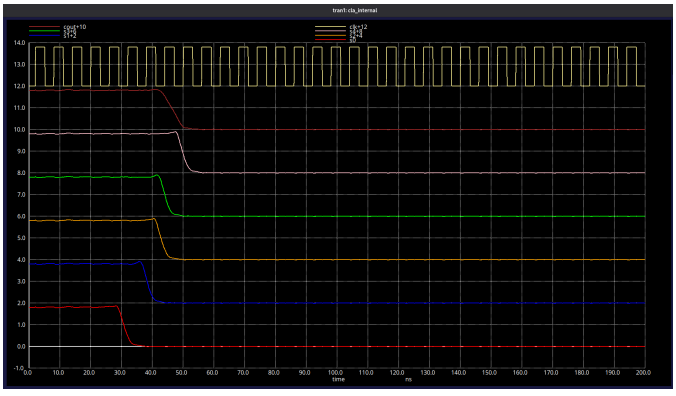

Fig. 42. Internal 5-bit CLA outputs (S0–S4 and Cout) showing combinational propagation delay.

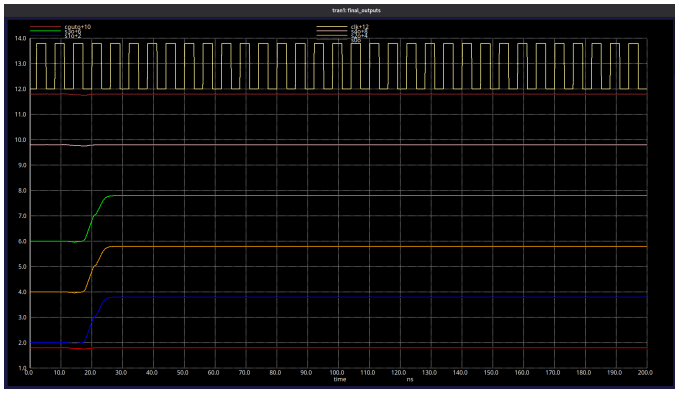

Fig. 43. Final registered outputs (S0o–S4o, Couto) captured by output D flip-flops.

Comparison of Pre-Layout and Post-Layout Timing: The pre-layout characterization of the NAND-only master–slave D flip-flop resulted in extremely small timing values, with $t_{\mathrm{clk}\to\mathrm{Q}} = 108.65$ ps, $t_{\mathrm{setup}} = 400$ ps, and $t_{\mathrm{hold}} = 450$ ps, as shown previously in Fig. 13. These values represent the idealized delays of the schematic-level design, where interconnect parasitics are not included.

In contrast, the post-layout simulation of the complete 5-bit pipelined CLA adder (which includes extracted parasitic capacitances from Magic) exhibits significantly larger delays. The manually extracted timing values were $t_{\mathrm{DFF,in}} \approx 8$ ns, $t_{\mathrm{CLA}} \approx 22$ ns, and $t_{\mathrm{DFF,out}} \approx 5$ ns, resulting in a total pipeline delay of approximately $t_{\mathrm{total}} \approx 35$ ns.

This large difference between pre-layout picosecond-scale delays and post-layout nanosecond-scale delays is expected. The layout introduces:

- additional wiring capacitance on every net,
- larger effective transistor loads due to fan-out,
- diffusion capacitances from series-connected devices,
- coupling and routing parasitics not present in the schematic.

Thus, while pre-layout results reflect the intrinsic device-level performance, post-layout simulation captures the realistic timing behavior of the fabricated circuit. The post-layout values therefore represent the true operational limits of the implemented 5-bit CLA adder.

IX. Verilog Implementation of the Pipelined 5-Bit CLA

The complete digital architecture combines a 5-bit Carry Look-Ahead (CLA) adder with master–slave D flip-flops to create a fully synchronous, clock-driven datapath. The objective of this Verilog implementation is to ensure that all inputs to the CLA block are sampled on the rising edge of the clock and that the generated sum and carry outputs are also registered before being driven out of the module. This creates a clean, one-cycle latency pipeline stage.

A. Registered Inputs

Each input bit of the operands $A[4:0]$ and $B[4:0]$, as well as the carry-in signal Cin, is connected to an individual master–slave D flip-flop. On every positive edge of the clock, these flip-flops capture the applied input values and generate the registered internal buses:

$$A_{\text{reg}}[4:0], \quad B_{\text{reg}}[4:0], \quad Cin_{\text{reg}}.$$

This ensures that the CLA block receives stable and time-aligned inputs.

B. Combinational CLA Block

The combinational module `cla5` implements the full 5-bit CLA logic, including bitwise generate and propagate calculations and the parallel carry computation. Using the registered inputs, the block produces the unregistered sum vector $S_{\text{comb}}[4:0]$ and the carry output $Cout_{\text{comb}}$ in the same cycle.

C. Registered Outputs

To maintain synchronous design discipline, the outputs of the CLA block are not directly driven to the external pins. Instead, each sum bit $S_{\text{comb}}[i]$ and the carry output $Cout_{\text{comb}}$ are passed through another stage of master–slave D flip-flops. This forms a complete clock-to-output pipeline boundary, producing:

$$S_{\text{out}}[4:0], \qquad Cout_{\text{out}}.$$

These signals represent the final registered outputs of the 5-bit CLA system.

D. Final RTL Structure

The resulting module `cla_final` integrates:

- 11 D flip-flops for input operand and carry registration,
- the combinational 5-bit CLA logic,
- 6 D flip-flops for output sum and carry registration.

In total, the design uses 17 master–slave D flip-flops surrounding the CLA block, forming a single-stage pipelined adder with one clock cycle latency.

Comparison of Pre-Layout and Post-Layout Timing: The pre-layout characterization of the NAND-only master–slave D flip-flop resulted in extremely small timing values, with $t_{\text{clk}\to\text{Q}} = 108.65$ ps, $t_{\text{setup}} = 400$ ps, and $t_{\text{hold}} = 450$ ps, as shown previously in Fig. 13. These values represent the idealized delays of the schematic-level design, where interconnect parasitics are not included.

In contrast, the post-layout simulation of the complete 5-bit pipelined CLA adder (which includes extracted parasitic capacitances from Magic) exhibits significantly larger delays. The manually extracted timing values were $t_{\text{DFF,in}} \approx 8$ ns, $t_{\text{CLA}} \approx 22$ ns, and $t_{\text{DFF,out}} \approx 5$ ns, resulting in a total pipeline delay of approximately $t_{\text{total}} \approx 35$ ns.

This large difference between pre-layout picosecond-scale delays and post-layout nanosecond-scale delays is expected. The layout introduces:

- additional wiring capacitance on every net,
- larger effective transistor loads due to fan-out,
- diffusion capacitances from series-connected devices,
- coupling and routing parasitics not present in the schematic.

Thus, while pre-layout results reflect the intrinsic device-level performance, post-layout simulation captures the realistic timing behavior of the fabricated circuit. The post-layout values therefore represent the true operational limits of the implemented 5-bit CLA adder.

Figure 44 shows the post-Verilog functional simulation waveforms for the complete `cla_final` module, which integrates the 5-bit Carry Lookahead Adder with master–slave D flip-flops at both the input and the output.

The signals `A_in[4:0]`, `B_in[4:0]`, and `Cin_in` are first captured on the rising edge of the clock. These registered values form the stable inputs to the combinational CLA logic. The resulting sum bits `S_comb[4:0]` and carry-out `Cout_comb` are then latched again into output D flip-flops on the next rising edge.

Because of this fully synchronous register-to-register structure, the design exhibits a one-clock-cycle latency: the outputs `S_out[4:0]` and `Cout_out` correspond to the inputs of the previous clock cycle. This expected pipelined behavior is clearly visible in the waveform, confirming:

- correct input latching on each rising edge,
- correct internal CLA computation for all applied vectors,
- correct output registration with a single-cycle delay,
- stable and glitch-free transitions across combinational boundaries.

The results verify that the Verilog implementation of the D-FF based CLA system matches the intended synchronous operation and correctly produces all sums and carry outputs.

Fig. 44. Functional simulation waveform of the final Verilog implementation of the 5-bit CLA with input and output D flip-flops. A clear one-cycle latency between the inputs and the outputs validates correct synchronous behavior.

X. FPGA Implementation

To validate the functional correctness of the 5-bit pipelined Carry Look-Ahead adder on real hardware, the design was also implemented on the AMD (Xilinx) Spartan-7 Boolean Board provided in the laboratory. This allows the behaviour of the CLA to be observed using physical switches and LEDs, ensuring that the RTL description works as expected outside simulation.

The Verilog module containing the input D flip-flops, the 5-bit CLA logic, and the output D flip-flops was programmed onto the Spartan-7 board using the Vivado tool. No architectural changes were required for FPGA deployment; the same RTL used for simulation was synthesized directly for the device.

Hardware Setup

The Boolean Board provides on-board I/O that allows direct interaction with the adder:

- Five switches for the inputs A_4–A_0,
- Five switches for B_4–B_0,
- One switch for the carry-in bit C_{in},
- Five LEDs to display the output sum bits S_4–S_0,
- One LED to display the final carry-out bit C_{out},
- A built-in clock that drives the pipeline flip-flops.

Functional Verification

After programming the FPGA, various combinations of inputs were applied using the switches. The corresponding outputs appeared on the LEDs and matched the expected sum and carry values in all tested cases. The pipelined behaviour was also visible: the inputs were registered on one clock edge, the CLA logic evaluated the sum, and the outputs were updated on the next clock edge.

This confirms that the 5-bit pipelined CLA adder functions correctly not only in simulation but also in real hardware.

FPGA Hardware Photographs

Photographs of the Spartan-7 Boolean Board during testing, along with the LED output patterns for representative input combinations, will be included at the end of the report.

XI. Conclusion

In this project, a fully pipelined 5-bit Carry Look-Ahead (CLA) adder was designed, implemented, and verified across multiple abstraction levels. The design flow began with the functional specification of the propagate–generate logic and the associated carry computation network. These blocks were combined with input and output D flip-flops to form a two-stage pipeline capable of capturing inputs, evaluating the sum, and producing registered outputs on successive clock edges.

Schematic-level simulations confirmed logical correctness and ideal timing. The design was then translated into physical layout using Magic, and post-layout parasitic extraction enabled accurate timing analysis. The post-layout waveforms demonstrated correct functionality of all subblocks and the complete pipelined architecture. Manual inspection of the timing results showed that the total propagation delay through the pipeline was approximately 35 ns, corresponding to a maximum operating frequency of around 28 MHz. As expected, these delays were significantly higher than the pre-layout picosecond-scale DFF characterizations due to interconnect capacitances and parasitic loading introduced during physical implementation.

To further validate correctness in real hardware, the RTL description of the pipelined CLA adder was programmed onto the AMD Spartan-7 Boolean Board. Inputs were applied using on-board switches, and the outputs were observed on LEDs. The FPGA implementation reproduced all expected results for a wide range of input combinations, demonstrating that the design operates correctly outside simulation and that the pipelined behaviour is preserved in hardware.

Overall, the project successfully demonstrated the complete digital VLSI design flow: from RTL modeling and pre-layout simulation, through physical layout and post-layout verification, to real-time hardware implementation. The close agreement between the simulated and observed results confirms the correctness and robustness of the 5-bit pipelined CLA adder. This work provides a strong foundation for understanding arithmetic unit design, timing closure, and the practical challenges involved in transitioning from schematic to layout to hardware.