DIGITAL PROJECT

Spring 2025

Participants (Team 5)

Names	**ID**
Ahmed Mustafa Abdulbadea	2200667
Mariam Amr Mohammad	2200704
Ibrahim Ahmed Ibrahim	2200785
Hana Walaa Abdelhakim	2200653
Hoor Wael Farouk Abdellah	2200298
Nour Ali Abdelrahman Mohamed	2200292
Neama Esam Mohamed	2200371
Eyad Mohamed Said Mohamed	2200436
Abdelrahman Ehab Elsayd	2200205
Miran Mohamed	2200604
Alaa Abdelhakeem Mahmoud	2200498

1. *Project Requirement*

- In this project we are required to design an ALU that can do both arithmetic and logic operations, as shown in Figure 1 , it's also required to place flipflops at the input and output of the ALU

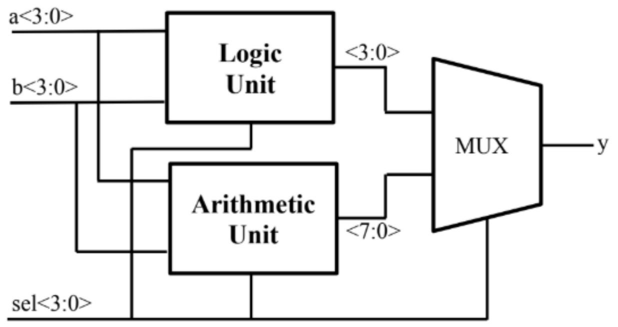


Fig. 1: ALU Block Diagram

Table 1: ALU Operations

Sel	Operation	Unit
0000	Increment a	Arithmetic
0001	Increment b	
0010	Transfer a	
0011	Transfer b	
0100	Decrement a	
0101	Multiply a and b	
0110	Add a and b	
0111	Subtract a and b (Assume a is always larger)	
1000	Complement a (1's complement)	Logic
1001	Complement b (1's complement)	
1010	AND	
1011	OR	
1100	XOR	
1101	XNOR	
1110	NAND	
1111	NOR	

2. Code for the ALU using Verilog

```verilog
module ArithmeticLogicUnit (
    input [3:0] a,
    input [3:0] b,
    input  [3:0] sel,
    output reg signed [7:0] y
);

reg signed [7:0] s1;
reg signed [7:0] s2;
reg signed [7:0] s3;
reg signed [7:0] s5;

always @(*) begin

   if (sel[3] == 1'b0) begin
        s3 = $signed(a);
        s5 = $signed(b);
        case(sel[2:0])
            3'b000: s1 = s3 + 1;
            3'b001: s1 = s5 + 1;
            3'b010: s1 = s3;
            3'b011: s1 = s5;
            3'b100: s1 = s3 - 1;
            3'b101: s1 = s3 * s5;
            3'b110: s1 = s3 + s5;
            3'b111: s1 = s3 - s5;
            default: begin s1 = 0; end
        endcase
        y = s1;
   end
    else begin
        case (sel[2:0])
        3'b000: s2 = ~a; // One's complement
        3'b001: s2 = ~b; // One's complement
        3'b010: s2 = a & b;
        3'b011: s2 = a | b;
        3'b100: s2 = a ^ b;
        3'b101: s2 = ~(a ^ b); // XNOR
        3'b110: s2 = ~(a & b); // NAND
        3'b111: s2 = ~(a | b); // NOR
        default: begin s2 = 0; end
        endcase
        y = s2;
    end
    // Select output from s1 or s2 based on sel[3]
end

endmodule
```

3. Testbench code testing all possible ALU operations

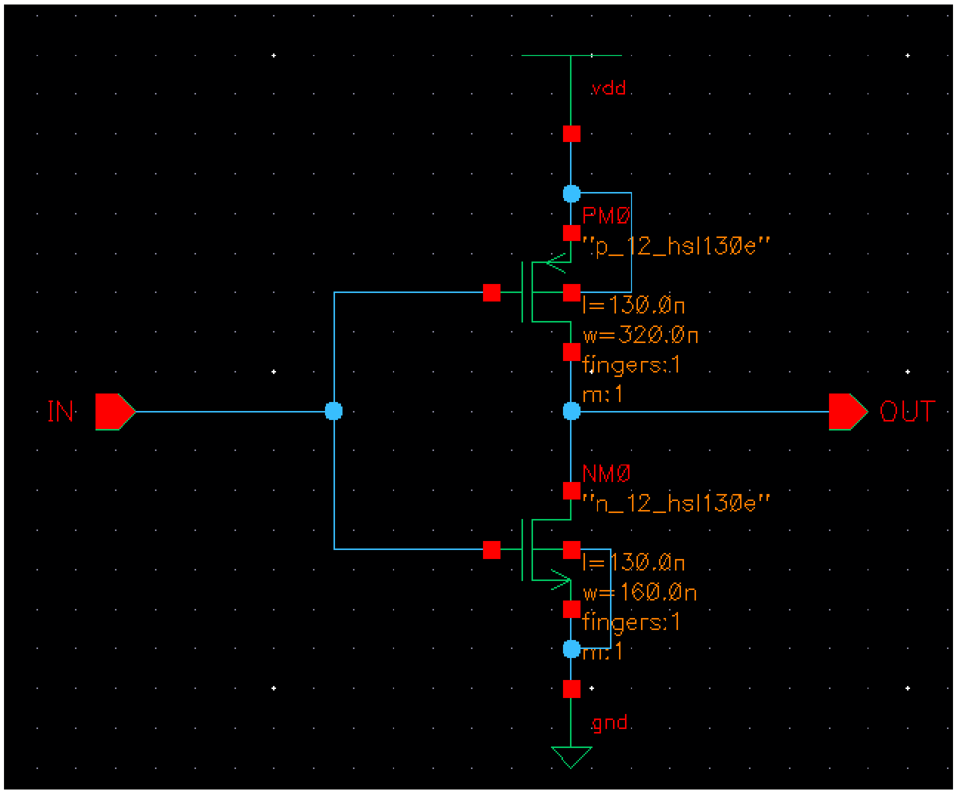
```
module ALU_tb ();
reg [3:0] a, b;
reg [3:0] sel;
wire [7:0] y;

ArithmeticLogicUnit A0 (
.a(a),
.b(b),
.sel(sel),
.y(y)
);
integer i;
initial
begin
for (i = 0; i < 16; i = i + 1)
begin
sel = i[3:0];  // assign s = 4-bit version of i
// Try a few different a/b combinations
a = 4'd1; b = 4'd5; #5;
a = 4'd2; b = 4'd4; #5;
a = 4'd3; b = 4'd3; #5;
a = -4'd4; b = 4'd2; #5;
a = -4'd5; b = 4'd1; #5;
end
end

initial
begin
$monitor("sel=%b | a=%d (decimal) a=%b (binary) | b=%d (decimal) b=%b
(binary) | result=%d (decimal) result=%b (binary)", sel, $signed(a), a,
$signed(b), b, $signed(y), y);
end
endmodule
```

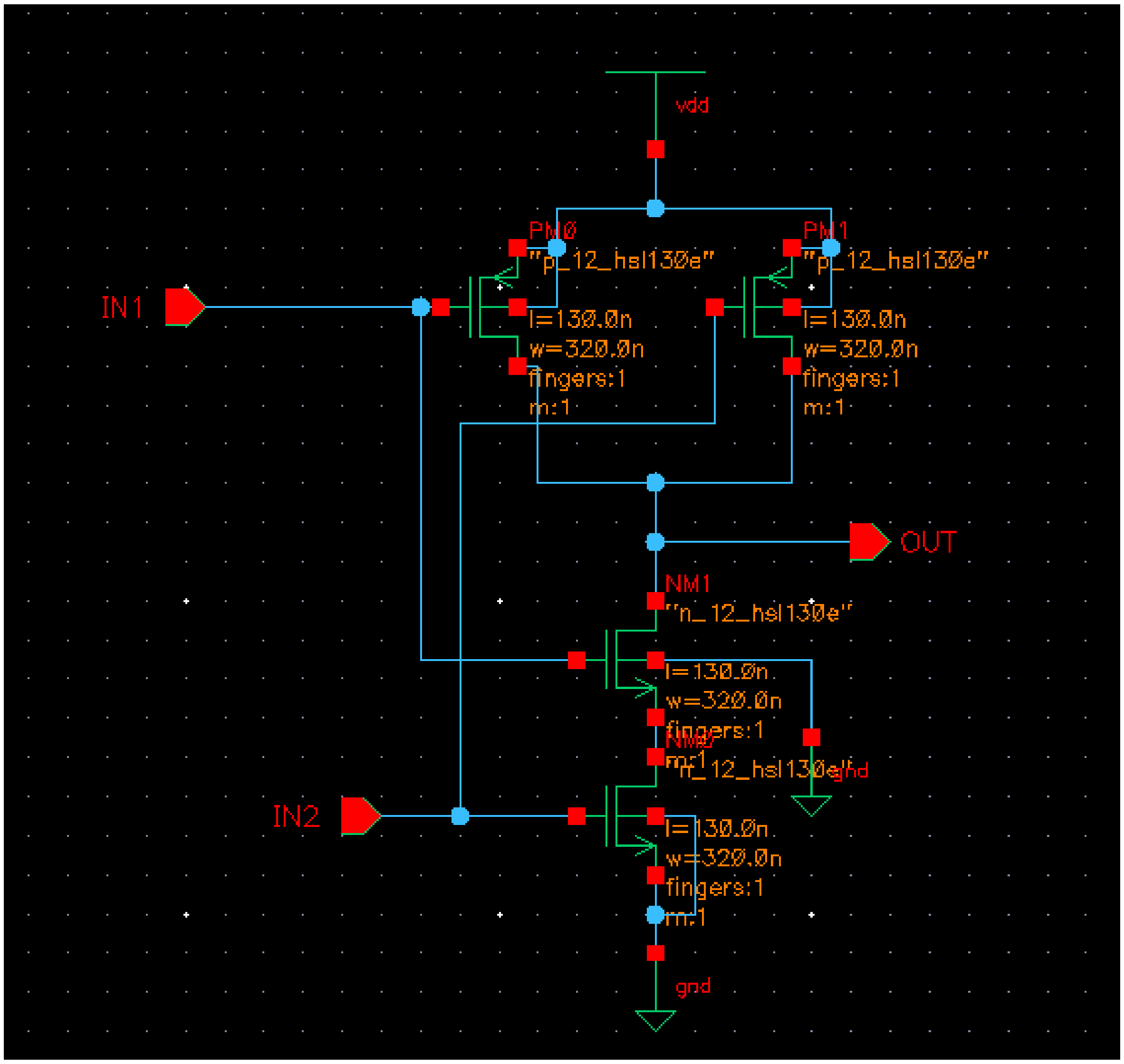
4. *Schematics on Cadence*

- In this section we will provide schematics for each component in the ALU done in cadence
- Using 130-nm CMOS technology.
- Inverter
  - Schematic



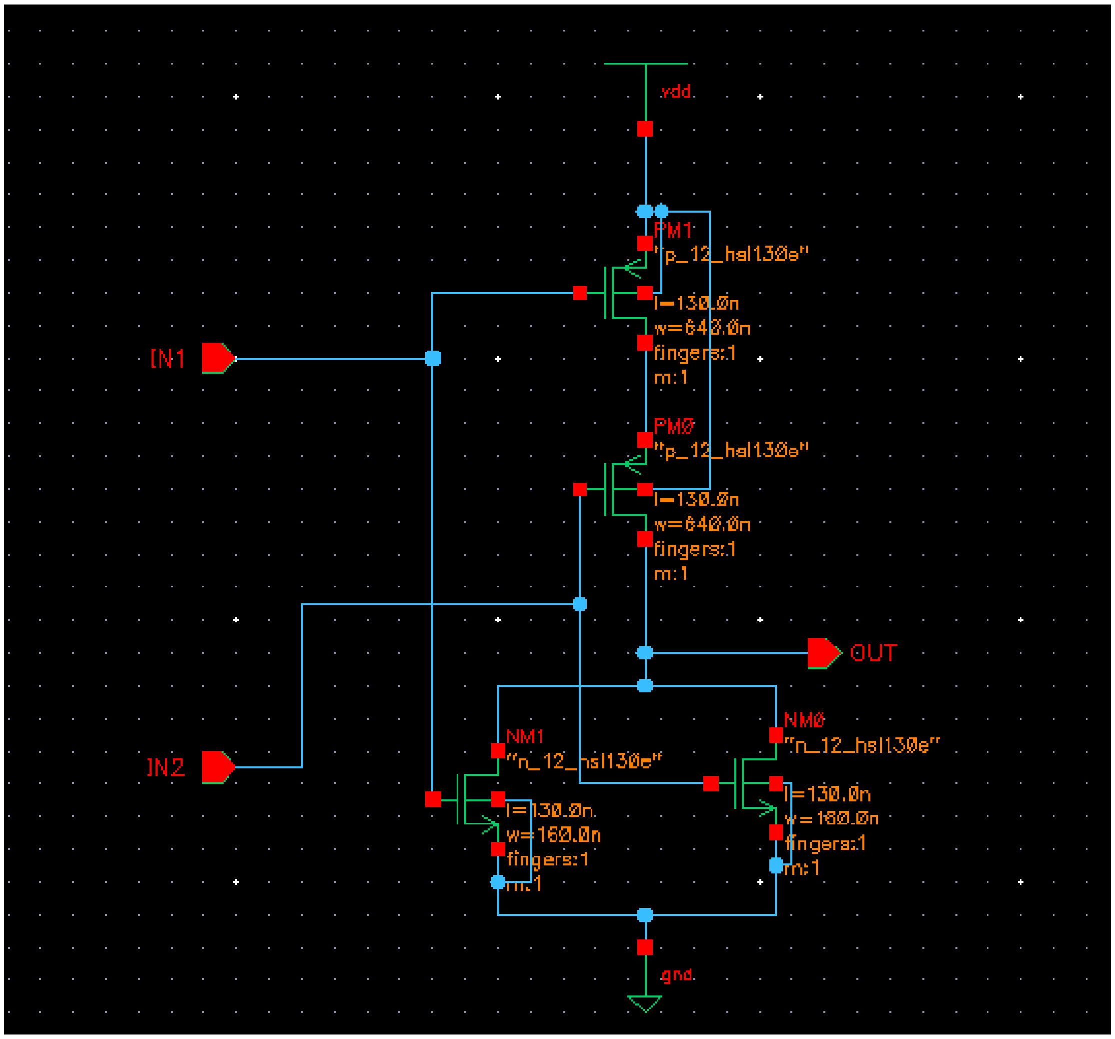
NAND gate

- Schematic



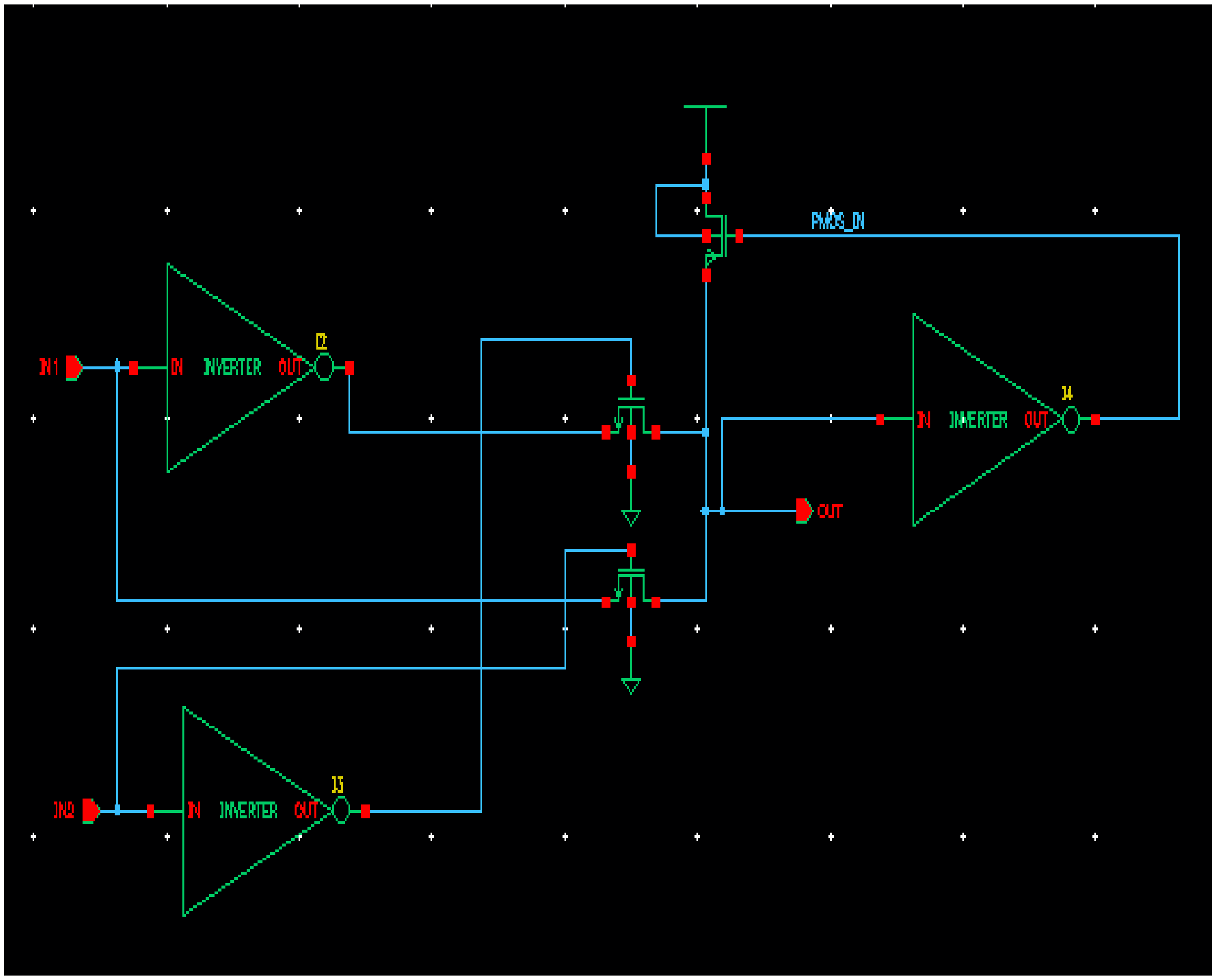
NOR gate

- Schematic



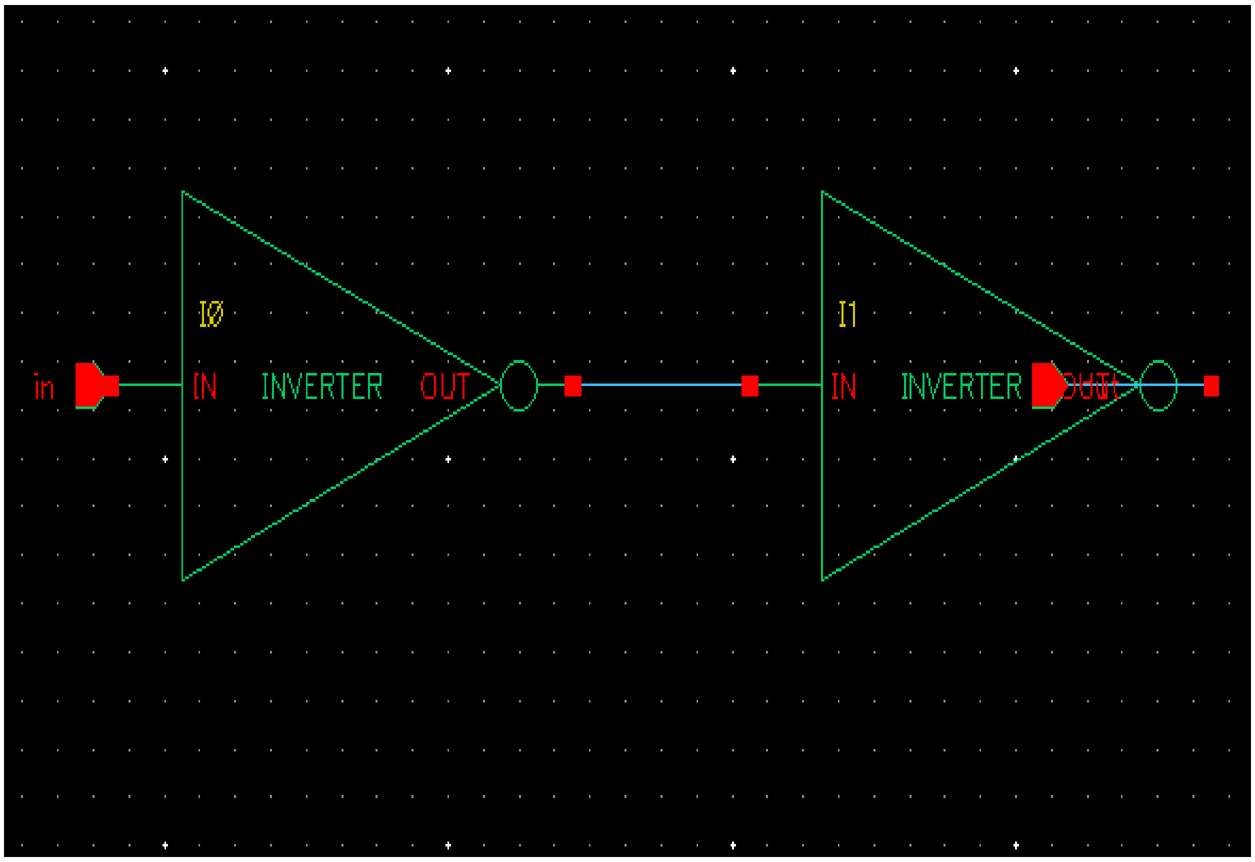
XNOR (PTL) gate

- Schematic

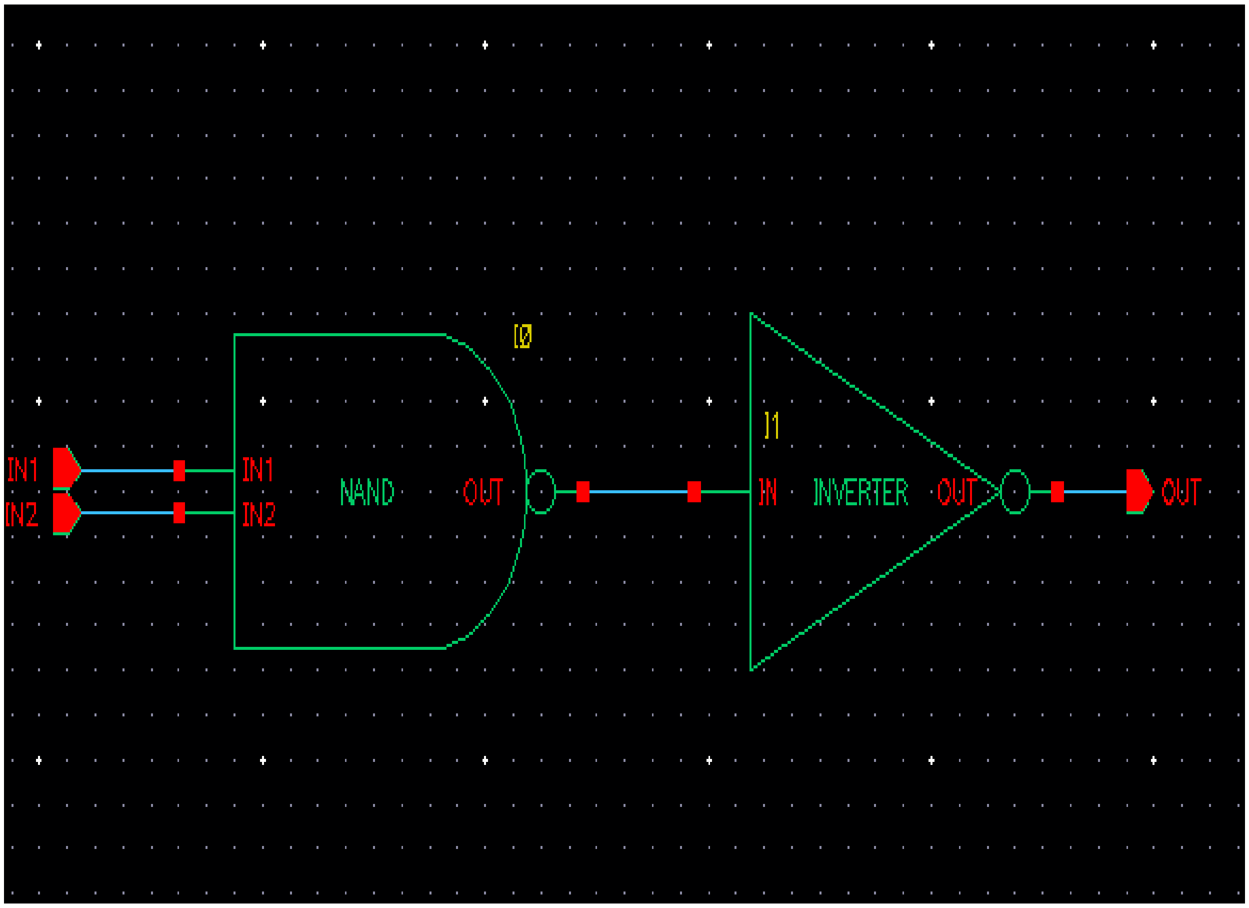


- Buffer

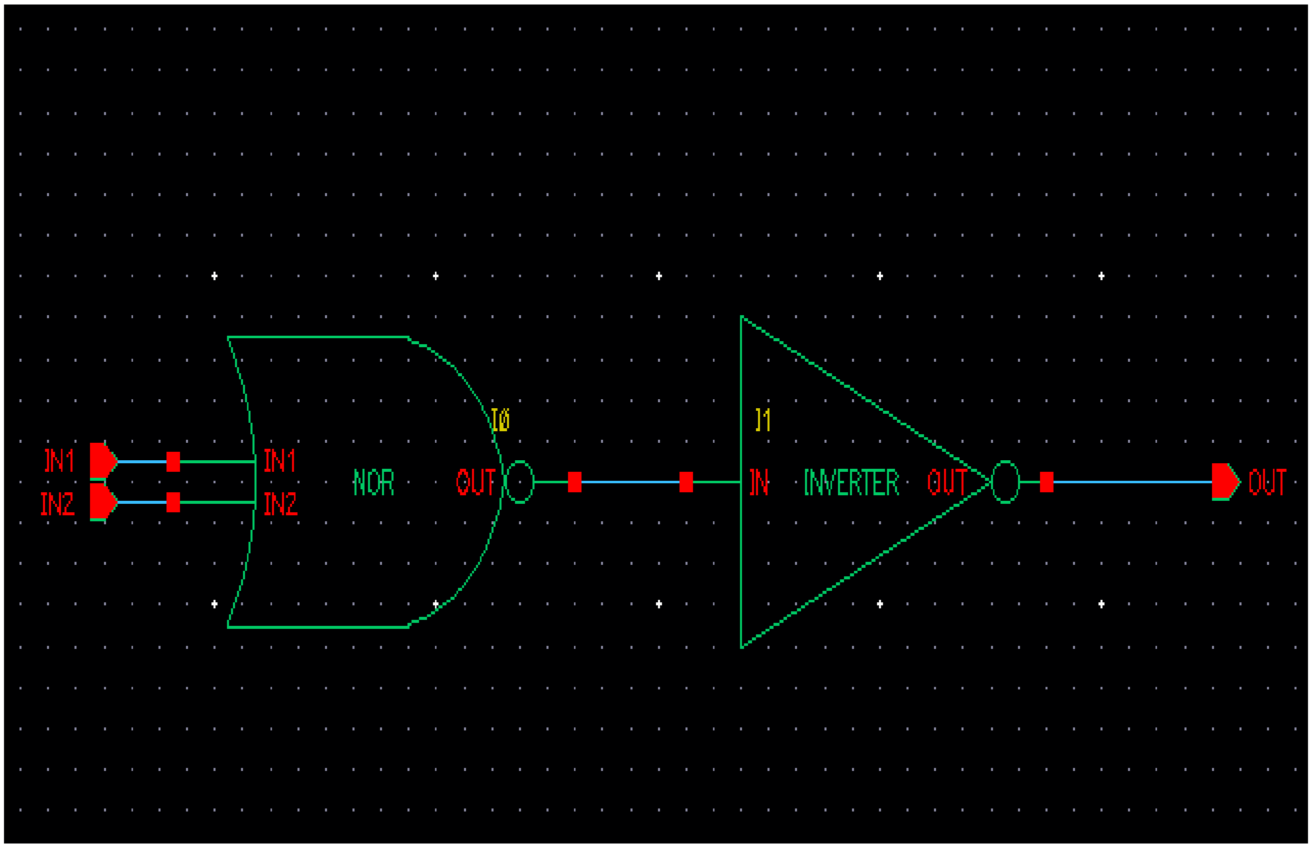
  - Schematic



- AND gate
  - Schematic

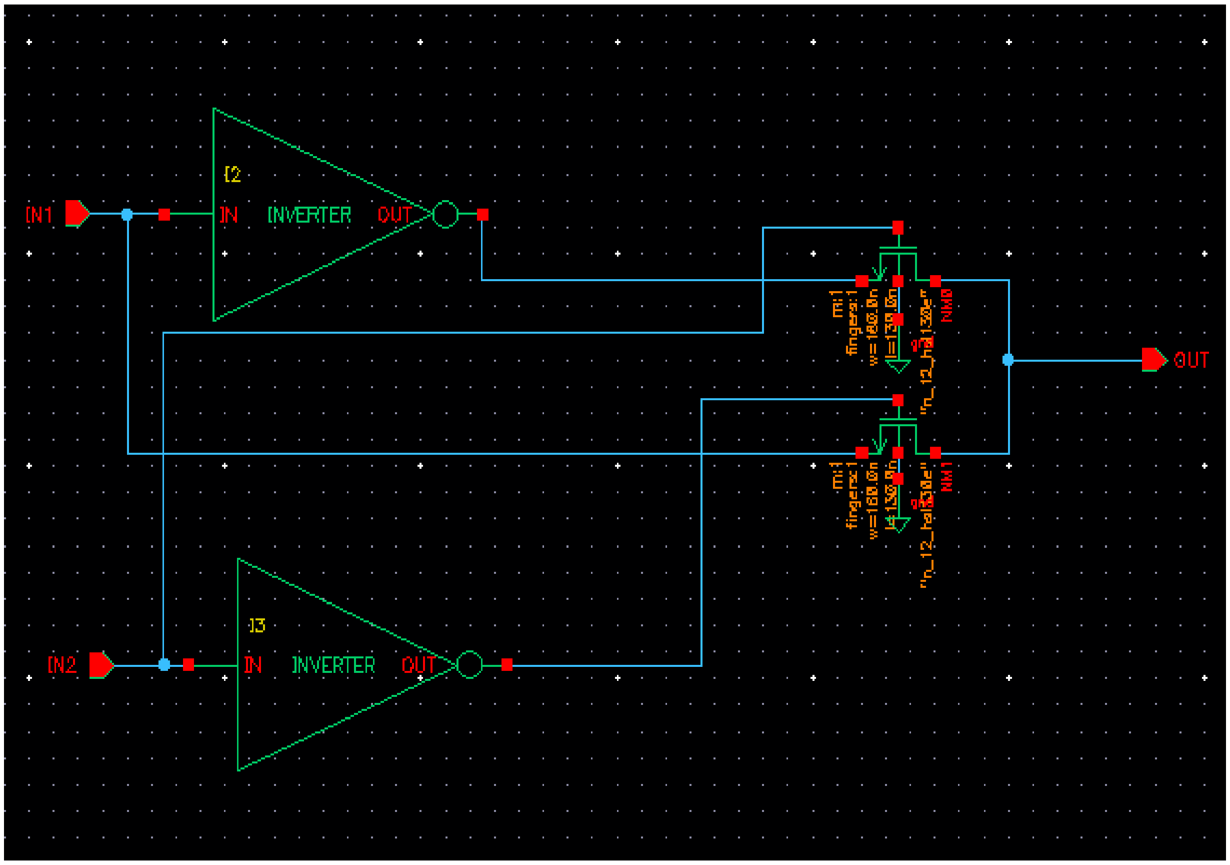


- OR gate
  - Schematic

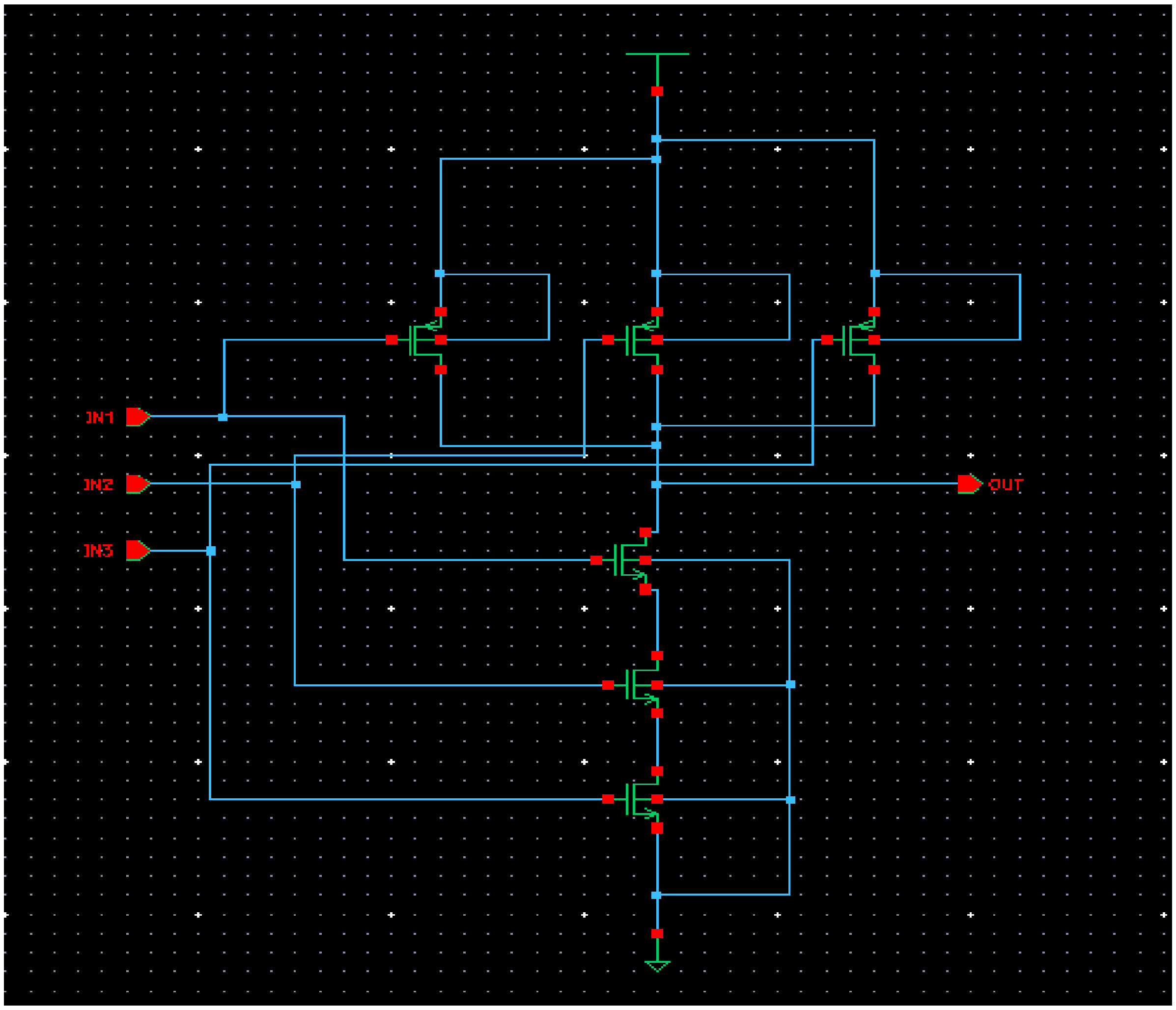


- XOR (PTL) gate

  - Schematic

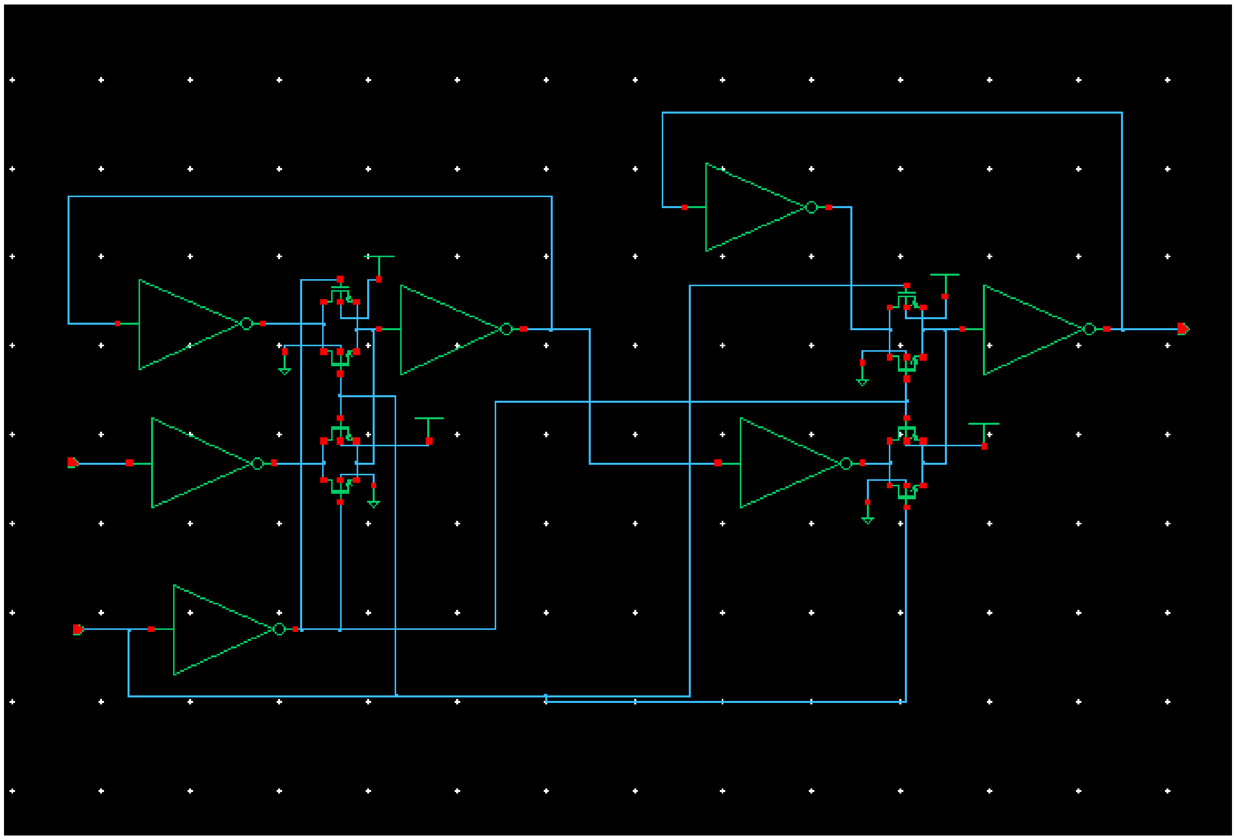


- 3 input NAND gate
  - Schematic

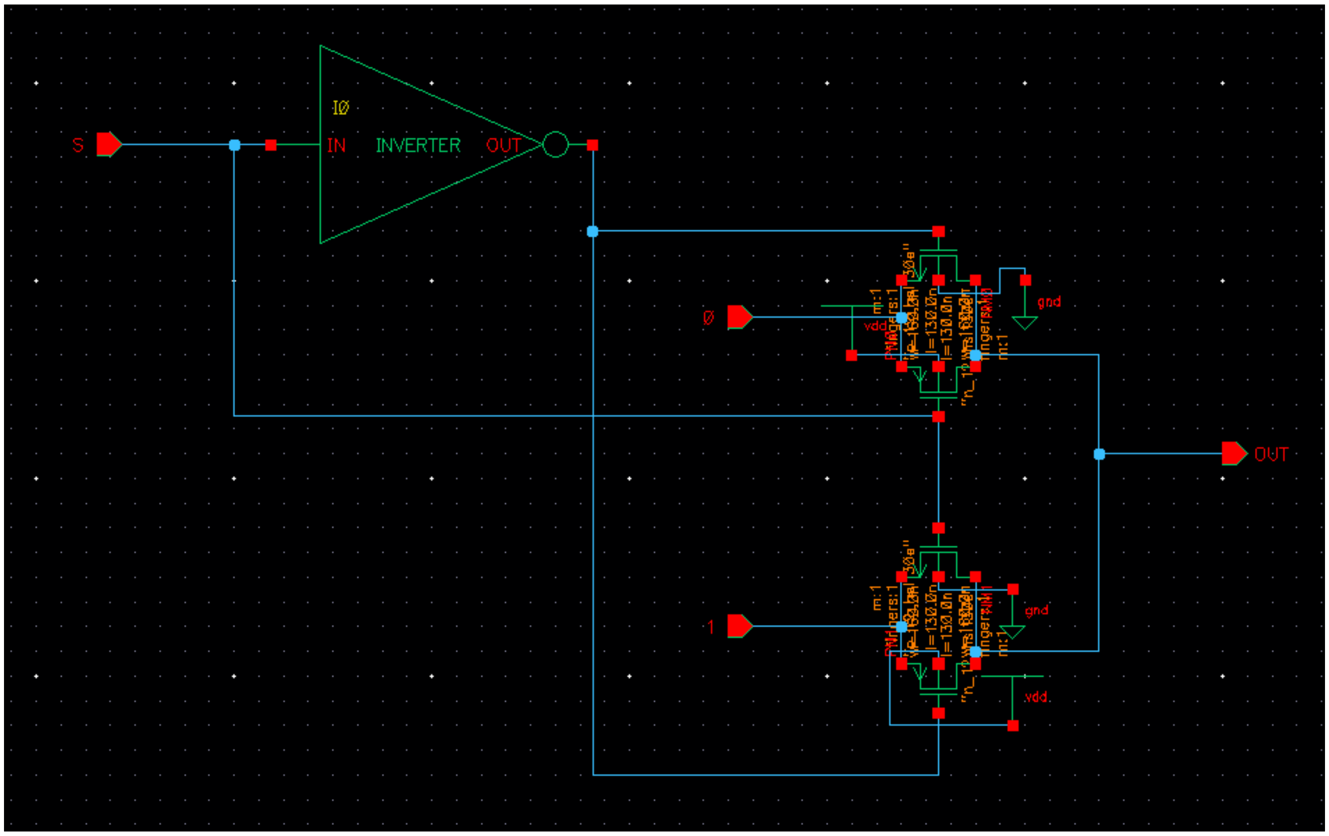


- Flip Flop

  - Schematic

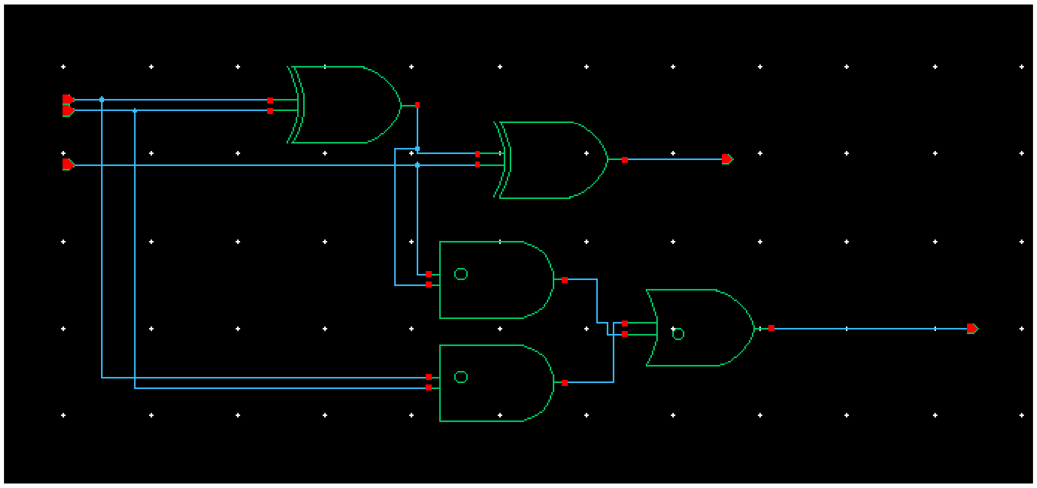


- TG 2X1 MUX
    - Schematic

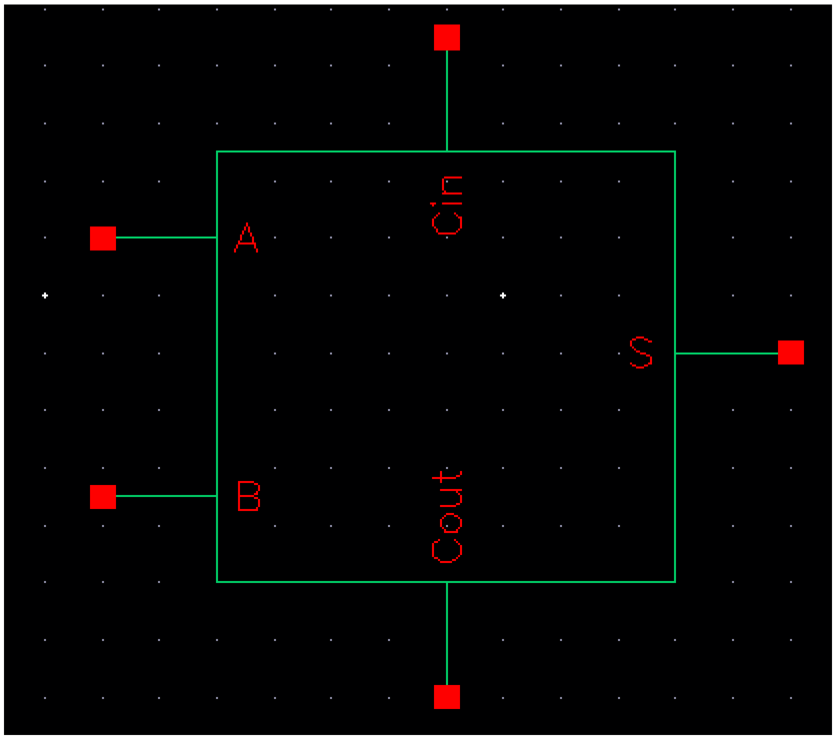


- Full adder

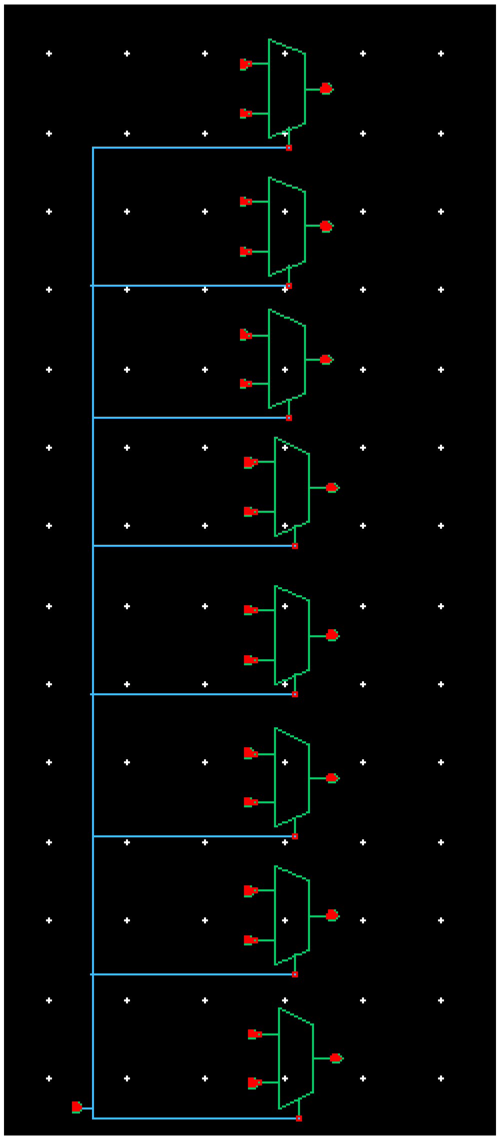
  - Schematic



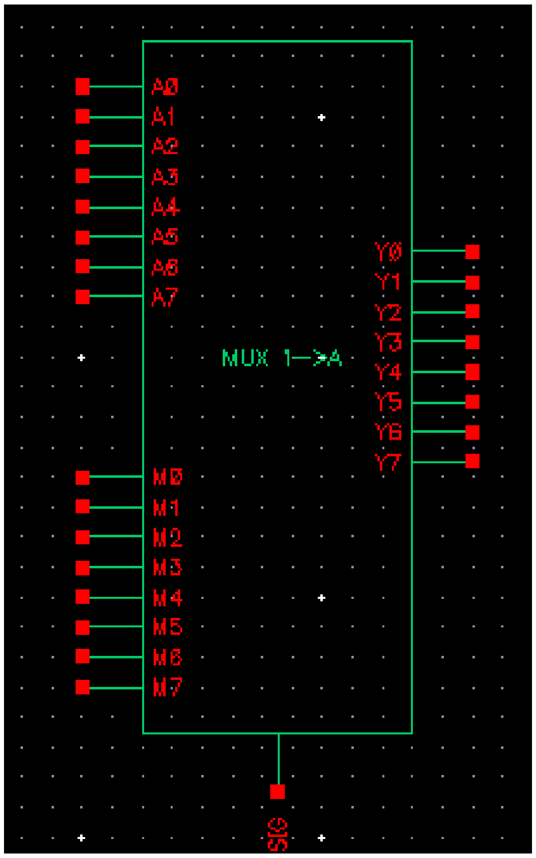
  - Symbol



- 8 input mux
  - Schematic

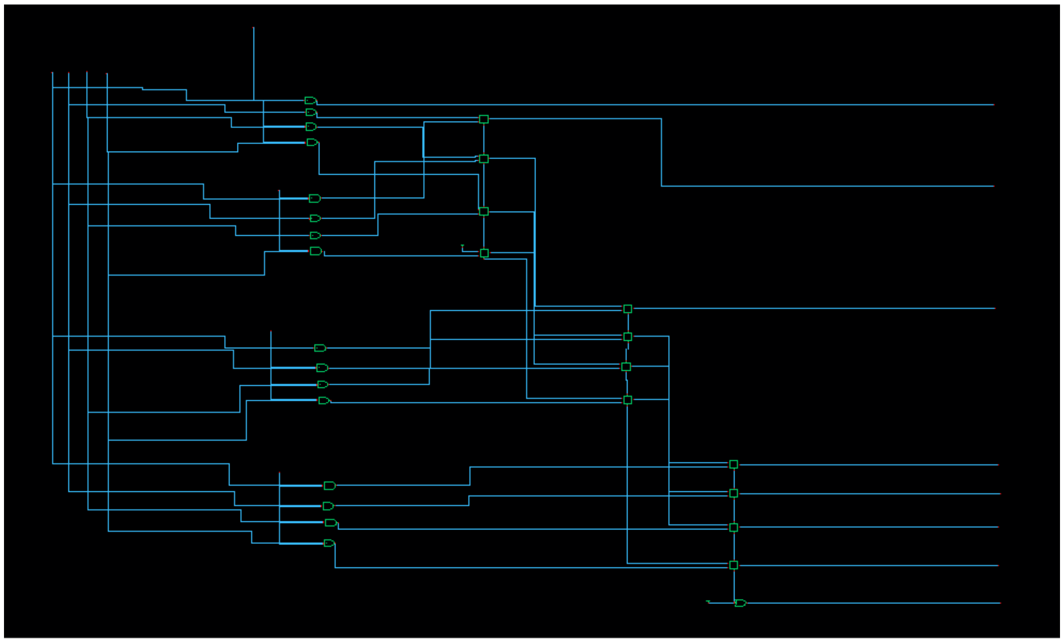


- Symbol

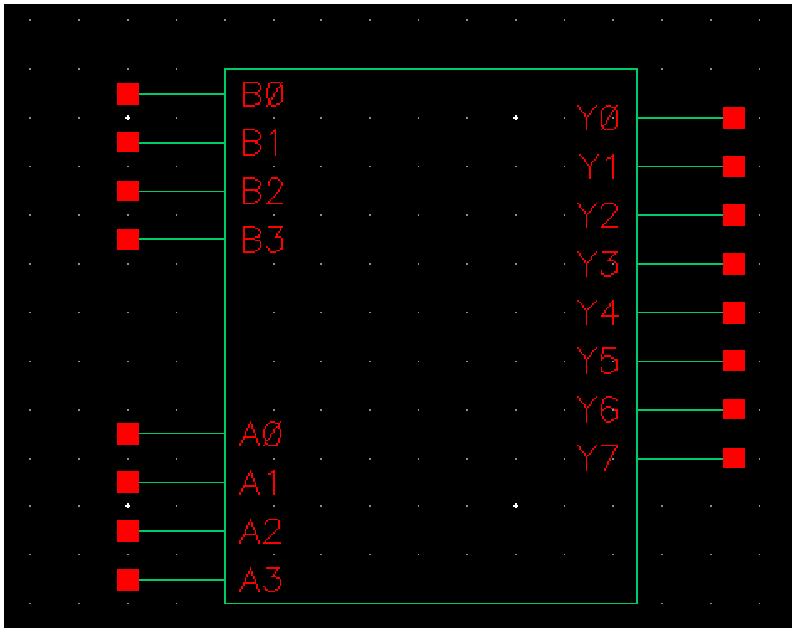

A0
A1
A2
A3
A4
A5
A6
A7
MUX 1->A
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
M0
M1
M2
M3
M4
M5
M6
M7
SIG

- Multiplier circuit

  - Schematic

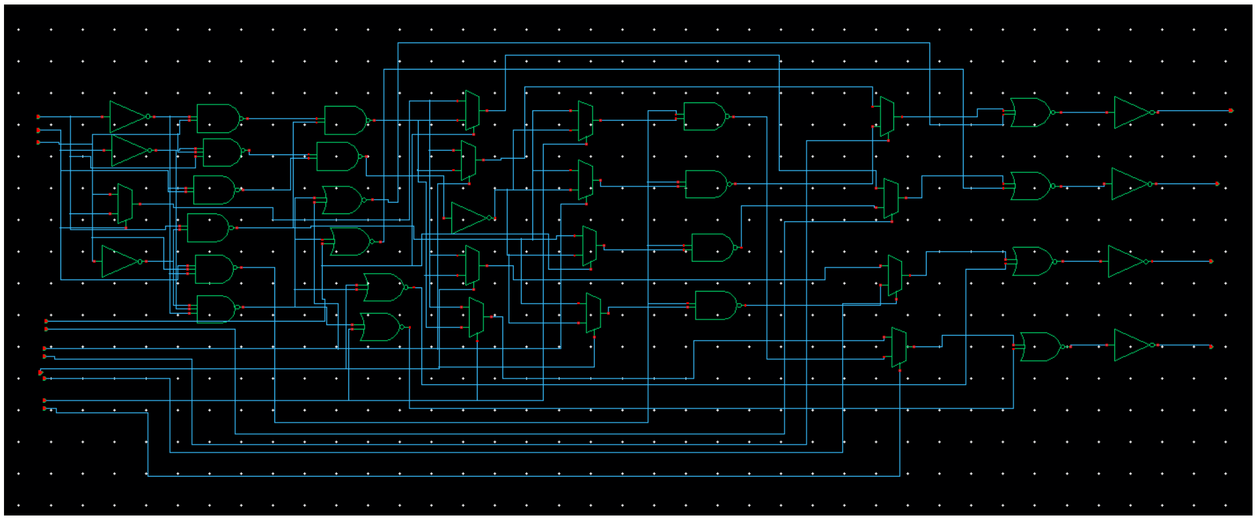


  - Symbol

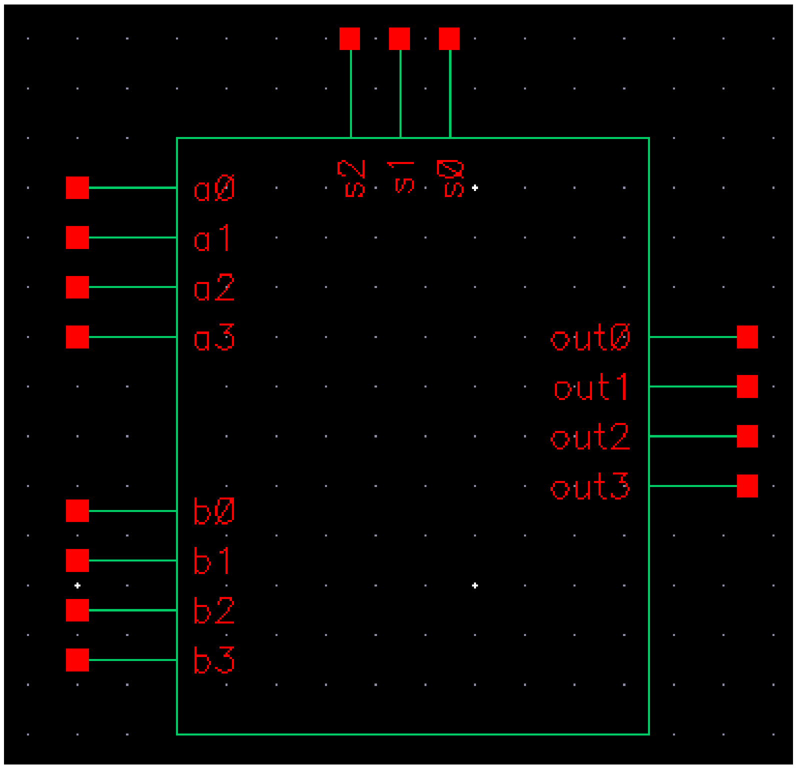


- Logic unit

  - Schematic

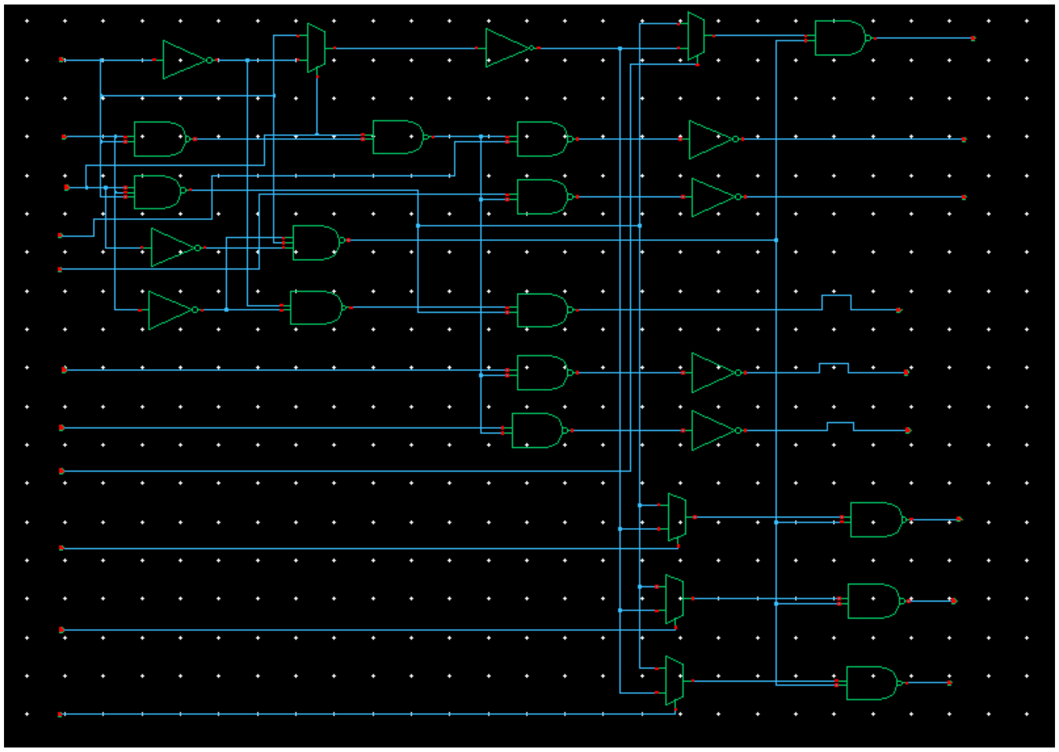


  - Symbol

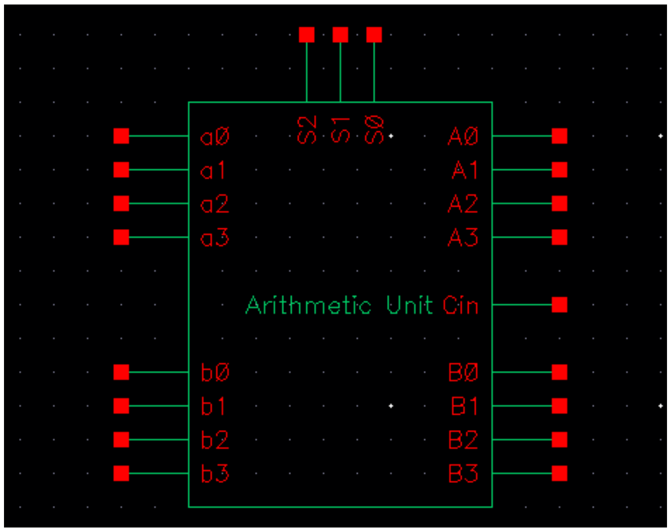


- Arithmetic unit without multiplier

  - Schematic

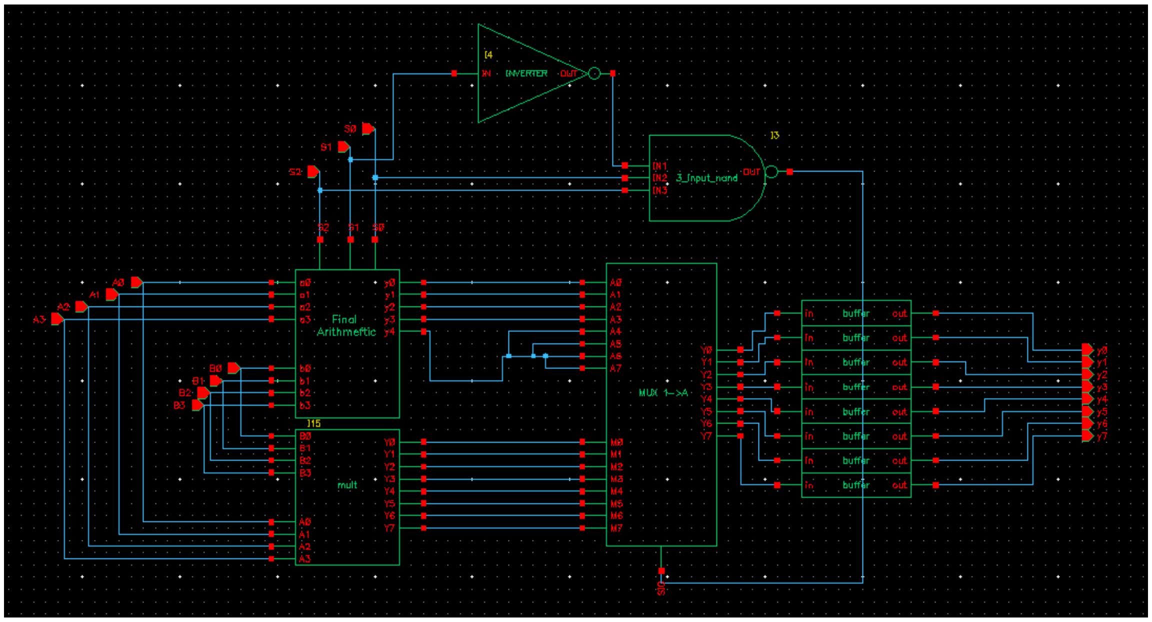


  - Symbol

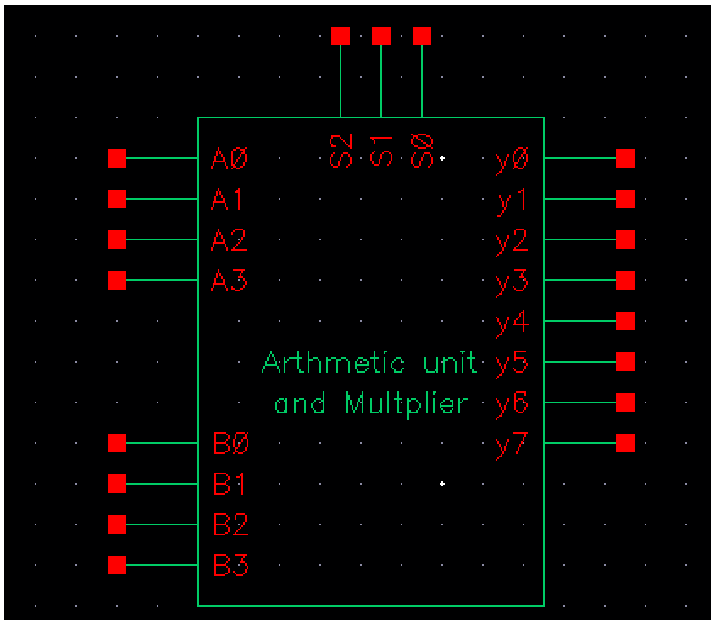


- Arithmetic unit with multiplier

  - Schematic

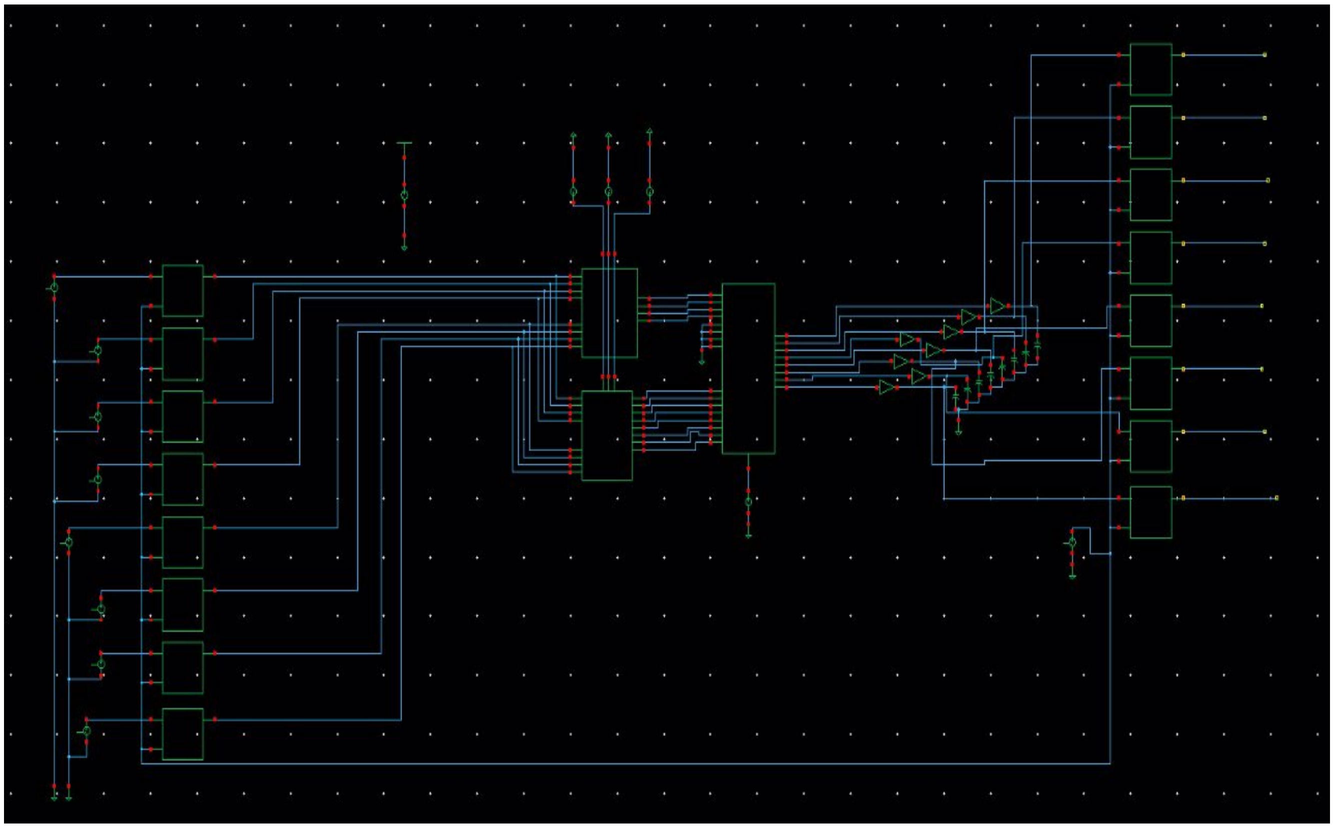


  - Symbol



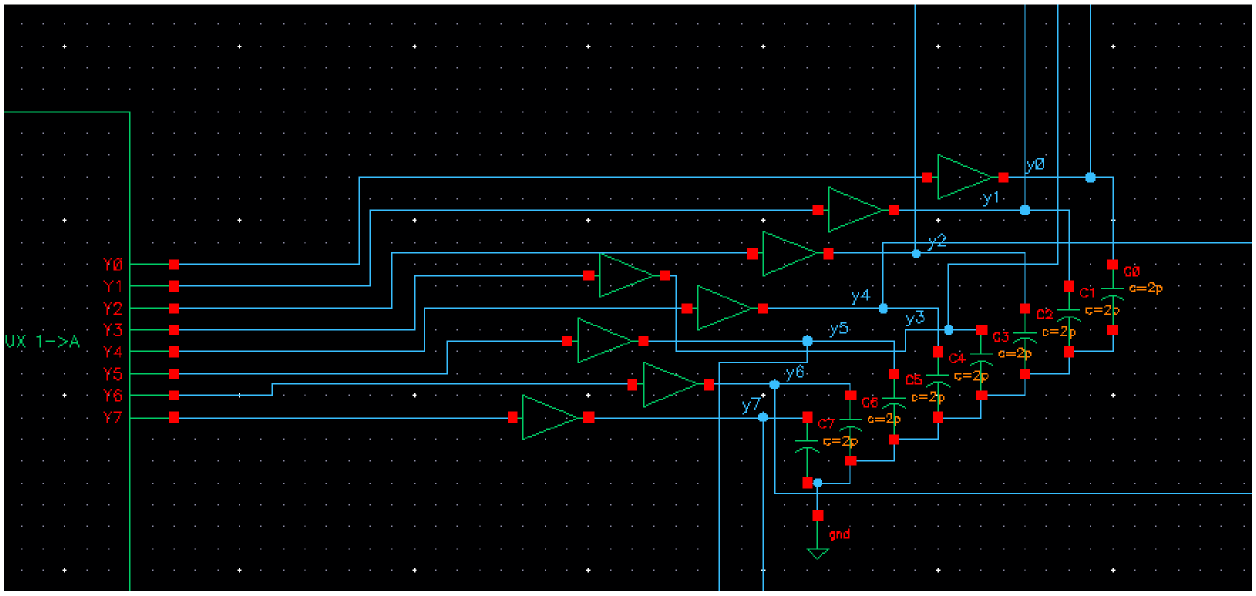
- ALU Final

  - Schematic



5. Capacitors connection (2pF)

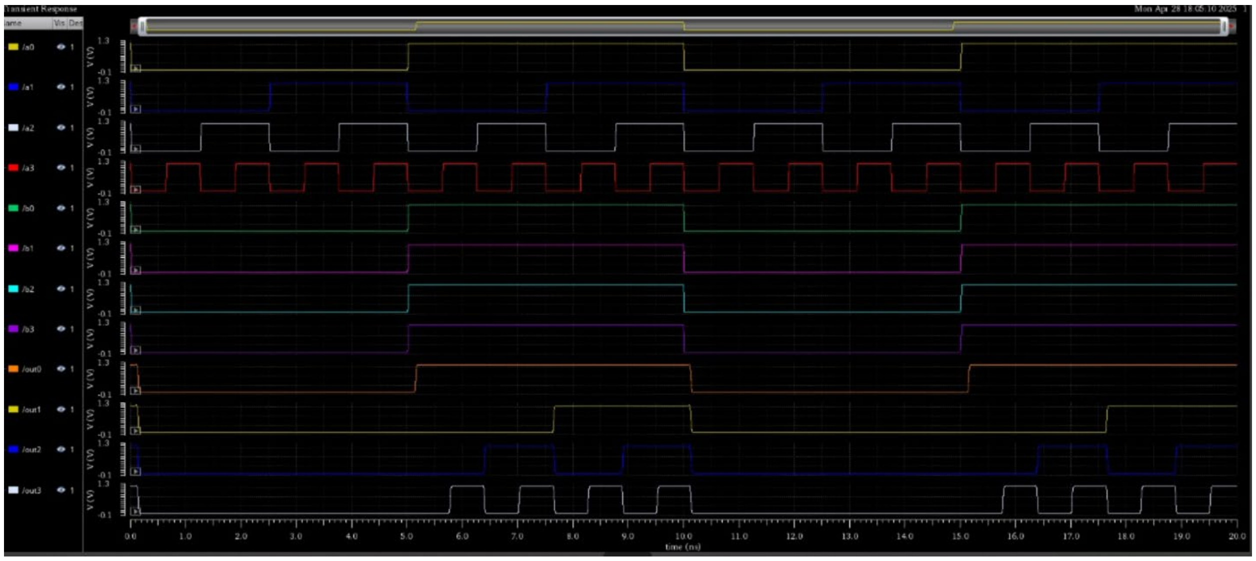
- We are specified to put a 2pF capacitor at the output of our ALU, it is big compared to the fan-out of our ALU, so we used a 4 Stage FO4 Buffer to drive the 2pF capacitor



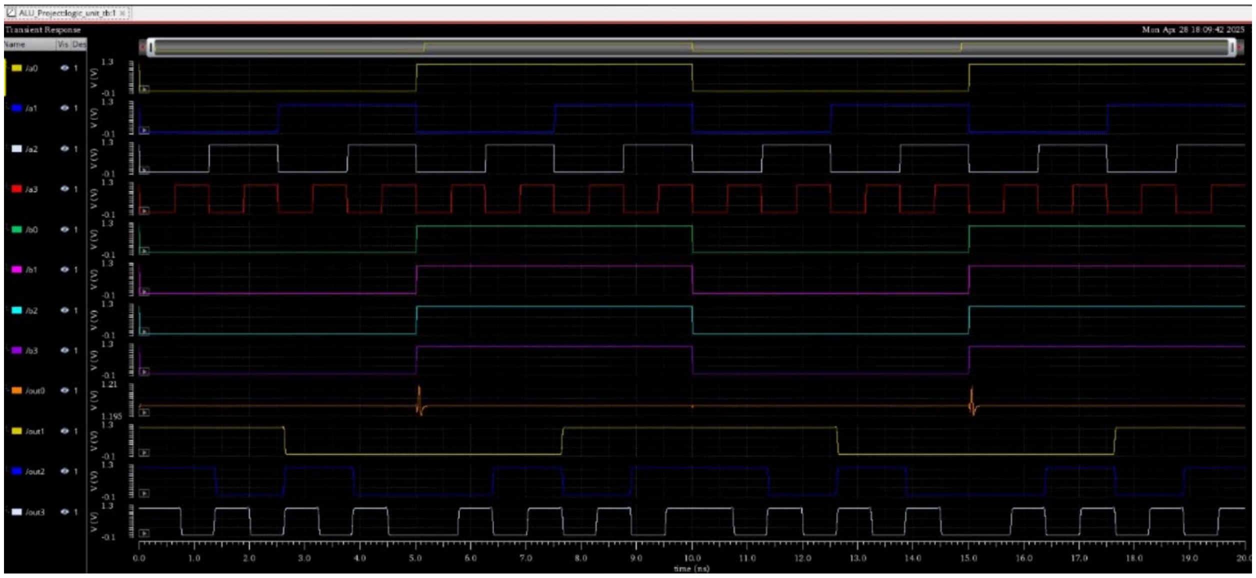
6. Output simulation in cadence

- Without flip flop (100MHz Input)

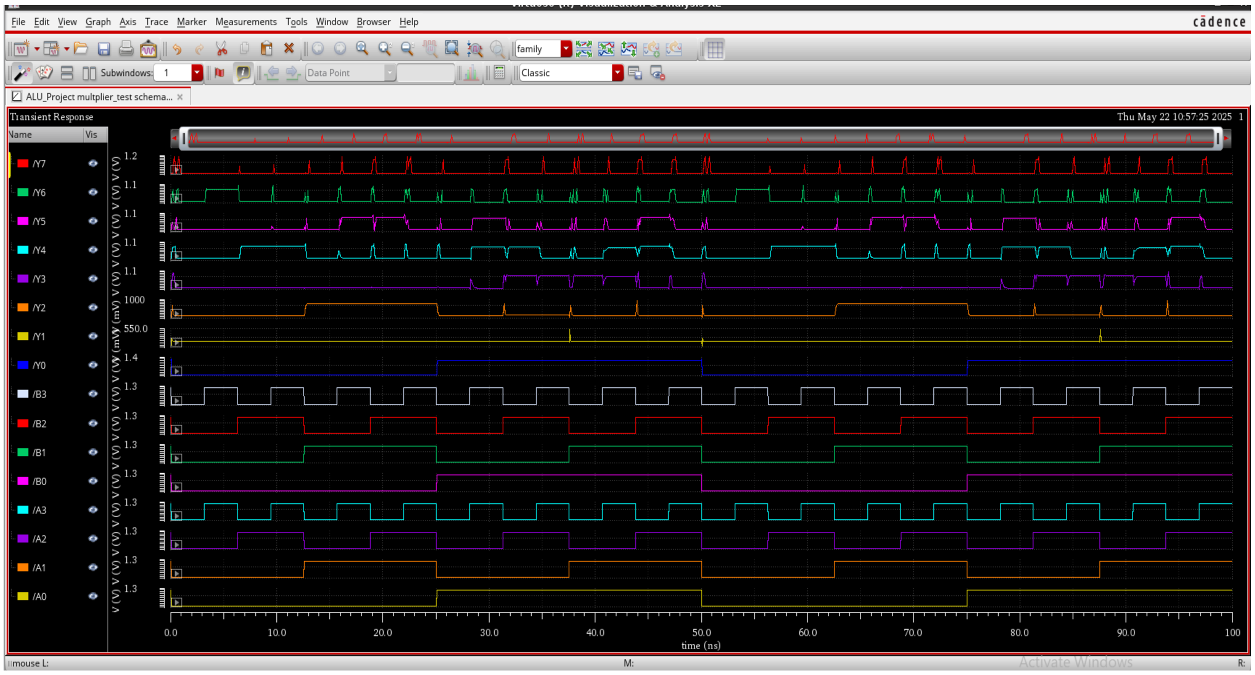
  - AND



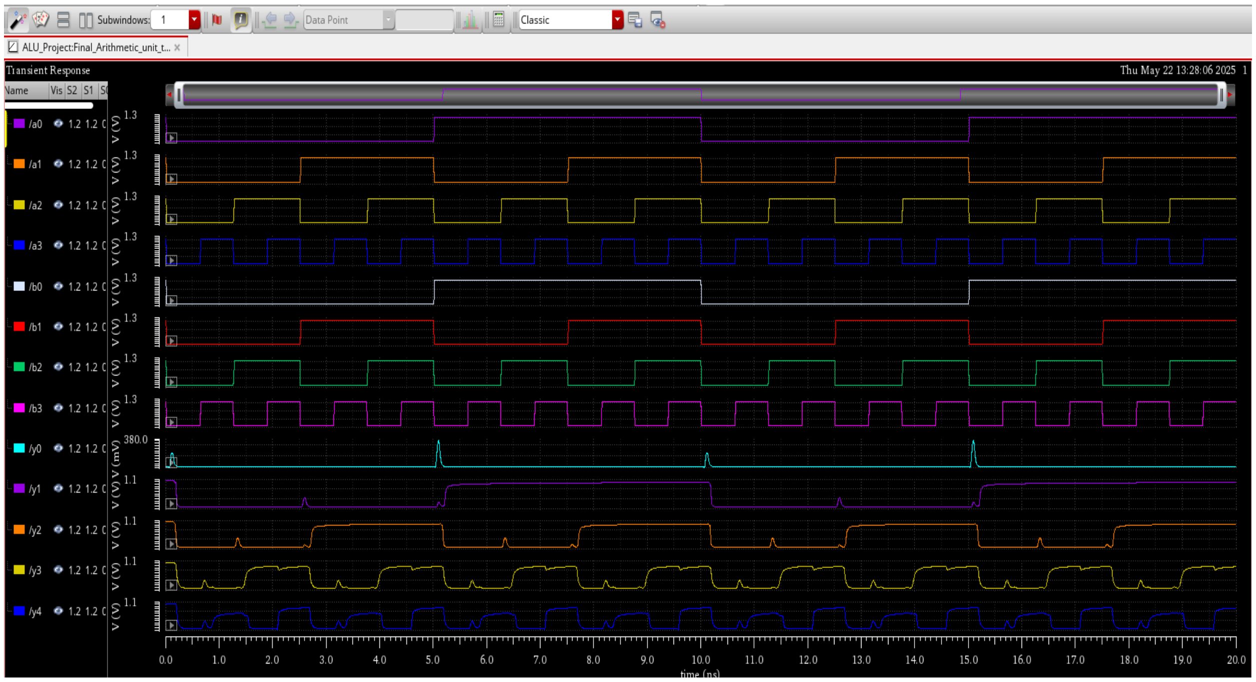
  - XNOR



- Multiplier

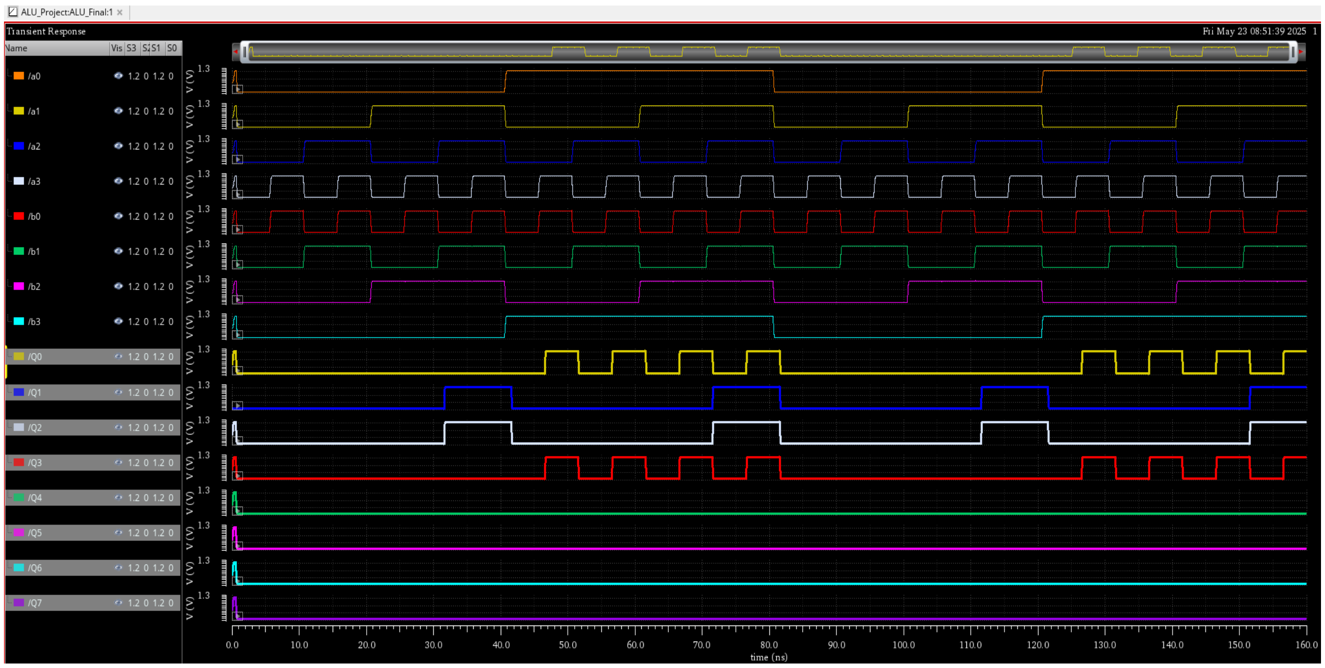

File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
ALU_Project multplier_test schema...
Transient Response
Thu May 22 10:57:25 2025
/Y7
/Y6
/Y5
/Y4
/Y3
/Y2
/Y1
/Y0
/B3
/B2
/B1
/B0
/A3
/A2
/A1
/A0
time (ns)

- Arithmetic operation (a+b) (110)

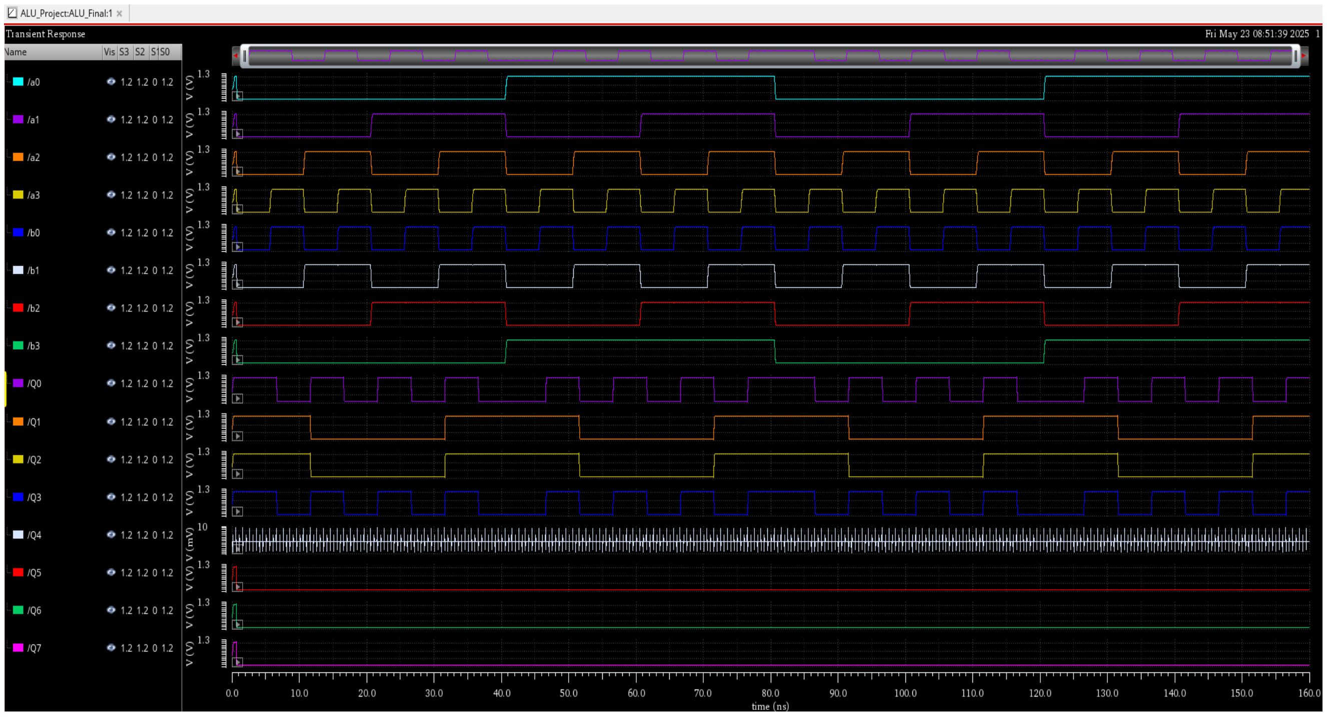


- With flipflop (100 MHz Input)

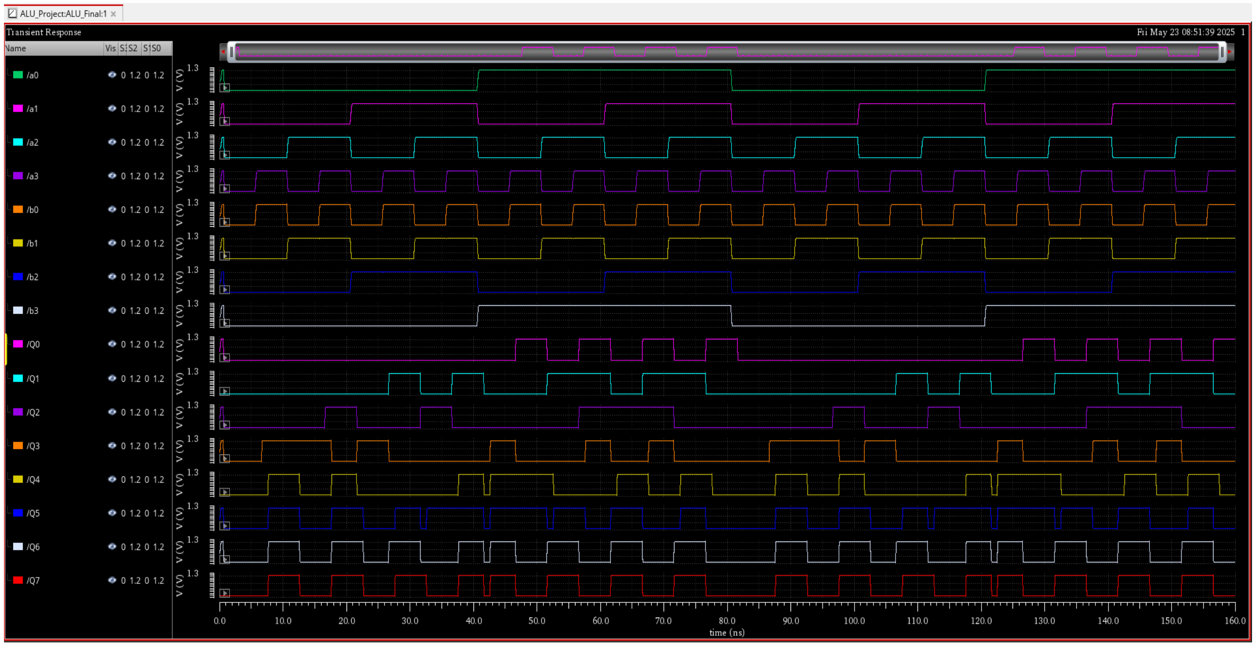
  - AND



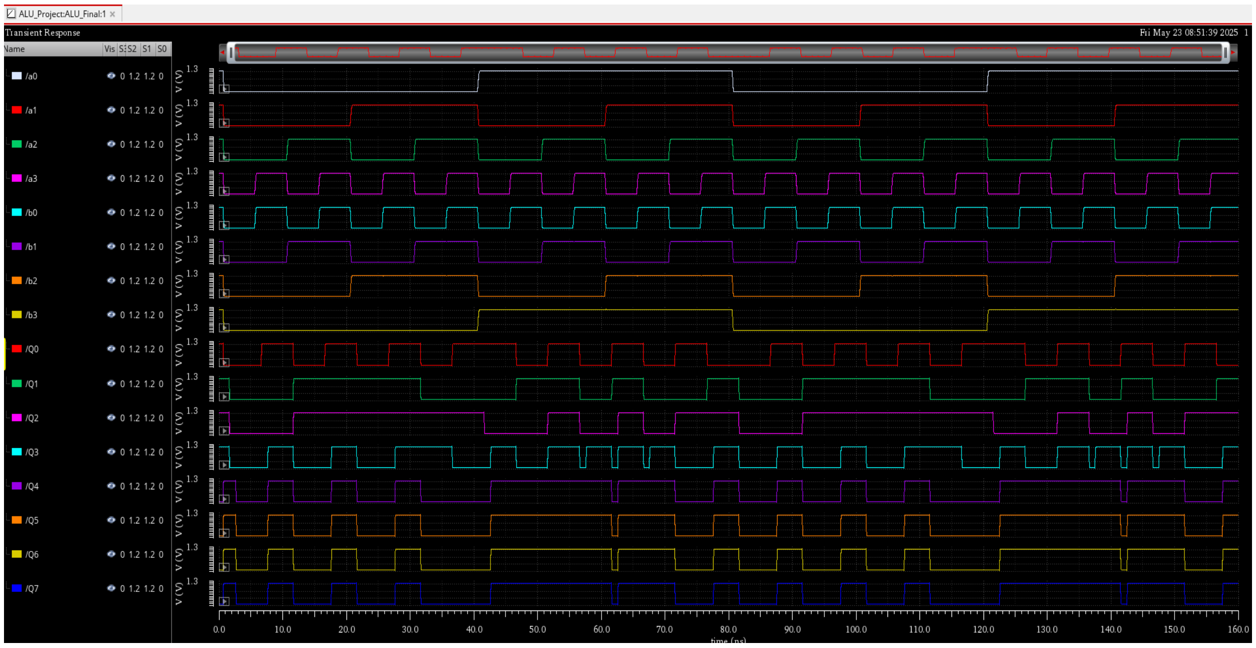
  - XNOR



- Multiplier



- Arithmetic operation (a+b) (110)



7. Circuit-Simulation (Delay and Power)

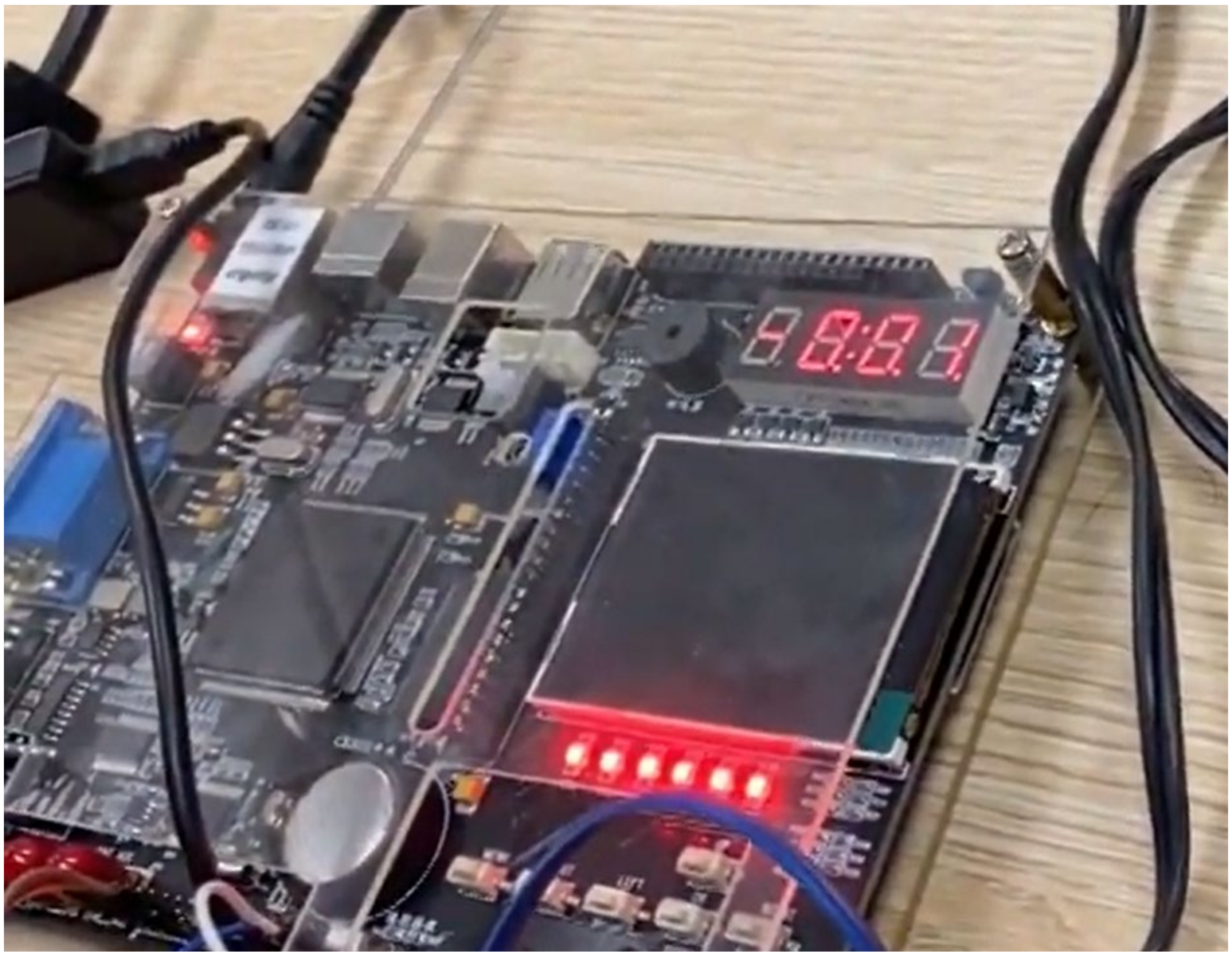
State	Power (mW)	Tplh (ns)	Tphl (ns)	Tp
0000	1.71	0.993	0.65	0.8215
0001	0.8445	0.7293	0.5869	0.6581
0010	1.938	0.9882	0.6515	0.81985
0011	0.8384	0.726	0.5391	0.63255
0100	1.685	0.91802	0.88833	0.90315
0101	1.984	1.122	1.1252	1.1236
0110	1.953	0.97891	0.65173	0.81532
0111	1.545	0.93477	0.778674	0.8567
1000	0.6958	0.56867	0.422404	0.4955
1001	0.6884	0.6021815	0.433931	0.518
1010	0.6405	0.64553	0.506397	0.5759
1011	0.5663	0.665767	0.483441	0.5746
1100	0.8237	0.65797	0.48676	0.5724
1101	0.8	0.653075	0.468203	0.5606
1110	0.592	0.674965	0.506581	0.5907
1111	0.5838	0.651575	0.466607	0.5591

- **This Data is taken with Input of 100MHz (Tperiod = 10ns)**
- **The CLK is set to 1ns**
- **The Highest Power and Delay are on the Multiplier (State 0101)**

8. *FPGA (Bonus)*

- **FPGA implementation of the semi-custom implementation**
  - **Video of FPGA**

    FPGA video



THANK

YOU