Carry Look Ahead Adder

Dhimant Bhuva
Electronic and Communication
Engineering
IIIT Hyderabad
Hyderabad, India
dhimant.bhuva@students.iiit.ac.in

***Abstract*—The Carry Look-Ahead (CLA) Adder is designed and implemented using 180nm CMOS technology, reducing propagation delay through hierarchical blocks. Validated using NGSPICE simulations and MAGIC layouts, the design is verified post-layout. A Verilog-based structural description enables FPGA implementation for hardware validation. Key performance metrics, including delay, maximum frequency, and power efficiency, demonstrate the CLA adder's high-speed and reliable operation.**

Keywords—CLA, NGSPICE, MAGIC, Delay, Verilog, FPGA

I. Introduction

Carry Look Ahead Adder (*CLA*) are improved version of Ripple Carry Adder (*RPA*). The primary disadvantage of RPA is carry propagation delay. Input bits are A_i, B_i and C_0 (i = 0 - 3). This problem is solved in CLA by pre-computing carries in parallel rather than waiting for "ripple" . Sequential dependency is removed by computing propagate(P_i) and generate(G_i) for each bit.

$P_i = A_i \wedge B_i$

$G_i = A_i \cdot B_i$

$C_{i+1} = G_i + P_i \cdot C_i$

Carry are computed in parallel as

$C_1 = G_0 + P_0 \cdot C_0$

$C_2 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$

$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$

$C_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$

Sum bits are calculated as

$S_i = P_i \wedge C_i$.

Also, positive edge triggered flip-flops are used to provide input and obtain output. If input bits are available before the rising edge of the clock, then output will be computed and present at the next rising edge of the clock.

II. Proposed Structure

For implementing CLA we need following logic gates :

- NOT
- OR – 2,3,4 and 5 input.
- AND – 2,3,4 and 5 input.
- XOR – 2 input.

All logic gates have been designed using Complementary CMOS logic style. D flip-flops are designed using True-Single-Phase-Clock (TSPC) topology.

Firstly, all inputs – bits a0 through a3 , b0 through b3 and c0 – are given as input to D flip-flop. Outputs of these flip-flops are then used to compute resulting sum bits, s0 through s3, and carry bit, c4, which itself are input to next stage of flip-flops. Overall design looks like 3 stage design , where 1st and 3rd stage are flip-flops and 2nd stage is logical block implementing CLA. All the gates have V_{DD} level of 1.8V.

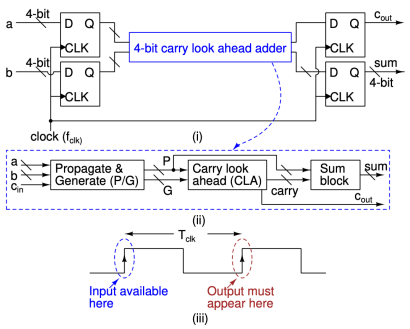


Figure 1 : 3 stage design overview

III. Topology and Mosfet Sizing

Minimum feature size is lambda, λ = 0.09µm. Minimum size not gate follows $W_P/ W_N = 20 λ/10 λ$.

A. *NOT gate*

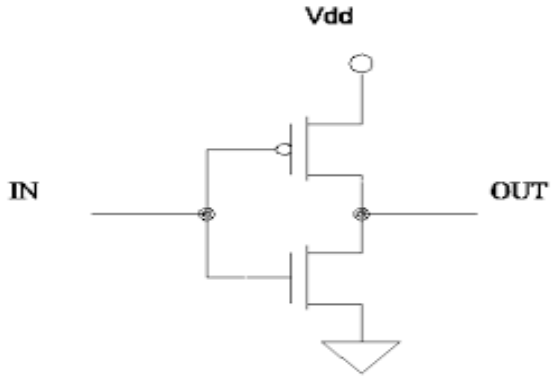


Figure 2 : NOT gate layout

- Width of NMOS, W_N = 900nm
- Width of PMOS , W_P = 1800nm

B. *OR gates*

I have implemented OR gates by cascading NOR gate with inverter. Same NOT gate has implemented as stated above. Sizing for NOR gate is provided below :

- *2 input*
  - Width of NMOS, W_N = 900nm
  - Width of PMOS , W_P = 3600nm
- *3 input*
  - Width of NMOS, W_N = 900nm
  - Width of PMOS , W_P = 5400nm
- *4 input*
  - Width of NMOS, W_N = 900nm
  - Width of PMOS , W_P = 7200nm
- *5 input*
  - Width of NMOS, W_N = 900nm
  - Width of PMOS , W_P = 9000nm

C. *AND gates*

Similarly, as OR gates , I have implemented AND gates by cascading NAND gate with inverter. Same NOT gate has implemented as stated above. Sizing for NAND gate is provided below :

- *2 input*
  - Width of NMOS, W_N = 1800nm
  - Width of PMOS , W_P = 1800nm
- *3 input*
  - Width of NMOS, W_N = 2700nm
  - Width of PMOS , W_P = 1800nm
- *4 input*
  - Width of NMOS, W_N = 3600nm
  - Width of PMOS , W_P = 1800nm
- *5 input*
  - Width of NMOS, W_N = 4500nm
  - Width of PMOS , W_P = 1800nm

D. *XOR gate*

- Width of NMOS, W_N = 1800nm
- Width of PMOS , W_P = 3600nm

E. *D flip-flop*

- Width of NMOS, W_N = 900nm
- Width of PMOS , W_P = 1800nm

IV. NGSpice Simulation

Using TSMC_180nm.txt technology file, I have implemented complete CLA circuit.

A. *NOT gate*

```
** NOT GATE **
.subckt inv y x vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
```

```
M1 y x vdd vdd CMOSP W={width_P}
L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P}
+AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M2 y x gnd gnd CMOSN W={width_N}
L={2*LAMBDA}
+ AS={5*width_N*LAMBDA}
PS={10*LAMBDA+2*width_N}
+AD={5*width_N*LAMBDA}
PD={10*LAMBDA+2*width_N}
.ends inv
```

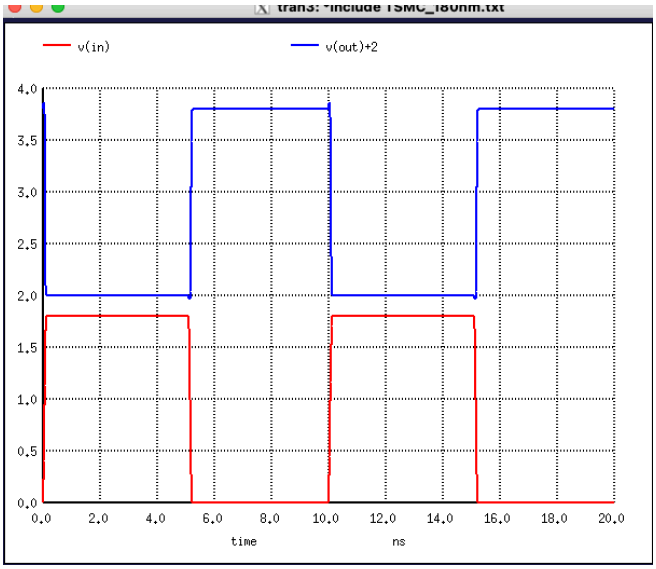


Figure 3 : NOT functionality

B. *OR gates*

- 2 input

```
.subckt or2 a b out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}

 * 2input NOR gate Wn = width_N Wp = 2*width_P *
M1 x a vdd vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA} PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA} PD={10*LAMBDA+2*2*width_P}
M2 nout b x vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA} PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA} PD={10*LAMBDA+2*2*width_P}

M3 nout a gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M4 nout b gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}

   *not gate*
M5 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M6 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends or2
```

Figure 4 : 2 input or gate netlist

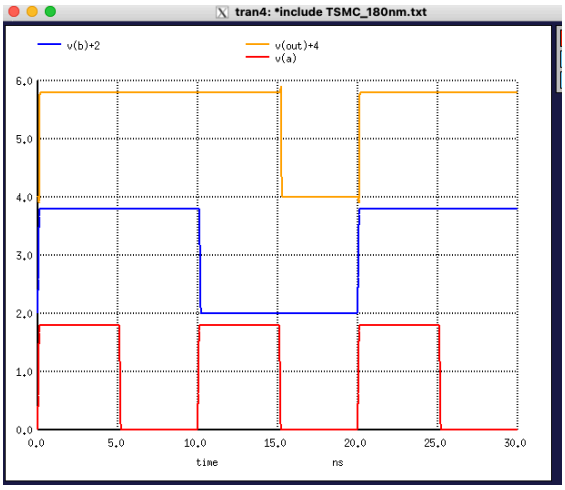


Figure 5 : 2input or gate functionality

- 3 input

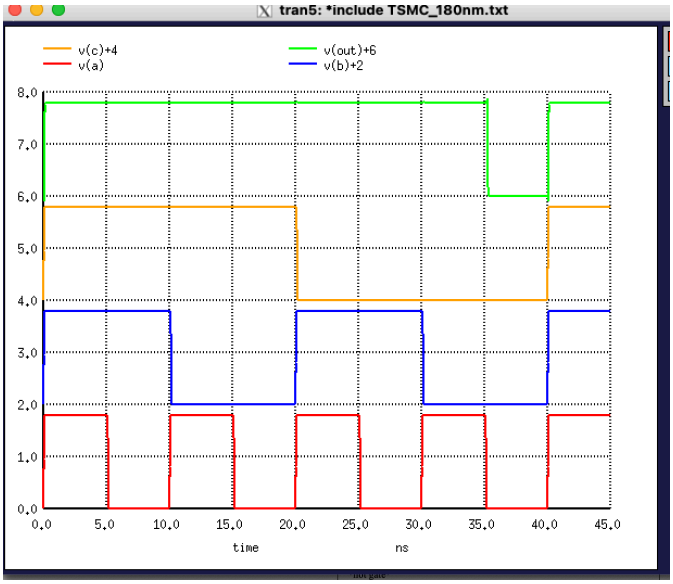
```
.subckt or3 a b c out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}

* 3input NOR gate Wn = width_N Wp = 3*width_P *
M1 x a vdd vdd CMOSP W={3*width_P} L={2*LAMBDA}
+ AS={5*3*width_P*LAMBDA}
PS={10*LAMBDA+2*3*width_P}
AD={5*3*width_P*LAMBDA}
PD={10*LAMBDA+2*3*width_P}
M2 y b x vdd CMOSP W={3*width_P} L={2*LAMBDA}
+ AS={5*3*width_P*LAMBDA}
PS={10*LAMBDA+2*3*width_P}
AD={5*3*width_P*LAMBDA}
PD={10*LAMBDA+2*3*width_P}
M3 nout c y vdd CMOSP W={3*width_P} L={2*LAMBDA}
+ AS={5*3*width_P*LAMBDA}
PS={10*LAMBDA+2*3*width_P}
AD={5*3*width_P*LAMBDA}
PD={10*LAMBDA+2*3*width_P}

M4 nout a gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M5 nout b gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M6 nout c gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}

*not gate*
M7 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M8 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends or3
```

Figure 6 : 3 input or gate netlist



- Figure 7 : 3 input or gate functionality

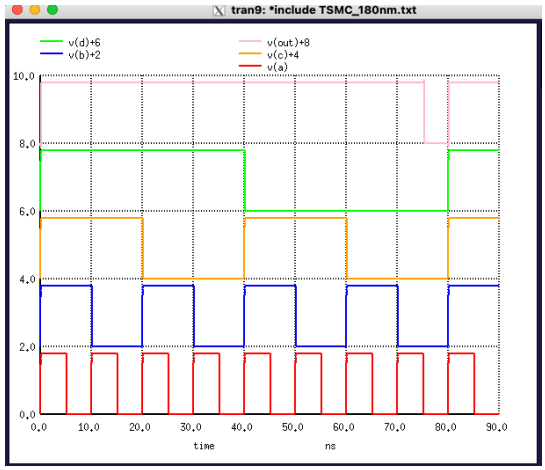


Figure 9 : 4 input or gate functionality

- 4 input

```
.subckt or4 a b c d out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
* 4input NOR gate Wn = width_N Wp = 4*width_P *
M1 x a vdd vdd CMOSP W={4*width_P} L={2*LAMBDA}
+ AS={5*4*width_P*LAMBDA}
PS={10*LAMBDA+2*4*width_P} AD={5*4*width_P*LAMBDA}
PD={10*LAMBDA+2*4*width_P}
M2 y b x vdd CMOSP W={4*width_P} L={2*LAMBDA}
+ AS={5*4*width_P*LAMBDA}
PS={10*LAMBDA+2*4*width_P} AD={5*4*width_P*LAMBDA}
PD={10*LAMBDA+2*4*width_P}
M3 z c y vdd CMOSP W={4*width_P} L={2*LAMBDA}
+ AS={5*4*width_P*LAMBDA}
PS={10*LAMBDA+2*4*width_P} AD={5*4*width_P*LAMBDA}
PD={10*LAMBDA+2*4*width_P}
M4 nout d z vdd CMOSP W={4*width_P} L={2*LAMBDA}
+ AS={5*4*width_P*LAMBDA}
PS={10*LAMBDA+2*4*width_P} AD={5*4*width_P*LAMBDA}
PD={10*LAMBDA+2*4*width_P}

M5 nout a gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M6 nout b gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M7 nout c gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M8 nout d gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}

*not gate*
M9 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M10 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends or4
```

Figure 8 : 4 input or gate netlist

- 5 input

```
.subckt or5 a b c d e out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}

* 5input NOR gate Wn = width_N Wp = 5*width_P *
M1 w a vdd vdd CMOSP W={5*width_P} L={2*LAMBDA}
+ AS={5*5*width_P*LAMBDA} PS={10*LAMBDA+2*5*width_P}
AD={5*5*width_P*LAMBDA} PD={10*LAMBDA+2*5*width_P}
M2 x b w vdd CMOSP W={5*width_P} L={2*LAMBDA}
+ AS={5*5*width_P*LAMBDA} PS={10*LAMBDA+2*5*width_P}
AD={5*5*width_P*LAMBDA} PD={10*LAMBDA+2*5*width_P}
M3 y c x vdd CMOSP W={5*width_P} L={2*LAMBDA}
+ AS={5*5*width_P*LAMBDA} PS={10*LAMBDA+2*5*width_P}
AD={5*5*width_P*LAMBDA} PD={10*LAMBDA+2*5*width_P}
M4 z d y vdd CMOSP W={5*width_P} L={2*LAMBDA}
+ AS={5*5*width_P*LAMBDA} PS={10*LAMBDA+2*5*width_P}
AD={5*5*width_P*LAMBDA} PD={10*LAMBDA+2*5*width_P}
M5 nout e z vdd CMOSP W={5*width_P} L={2*LAMBDA}
+ AS={5*5*width_P*LAMBDA} PS={10*LAMBDA+2*5*width_P}
AD={5*5*width_P*LAMBDA} PD={10*LAMBDA+2*5*width_P}

M6 nout a gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M7 nout b gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M8 nout c gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M9 nout d gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M10 nout e gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}

*not gate*
M11 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M12 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends or5
```

Figure 10 : 5 input or gate netlist

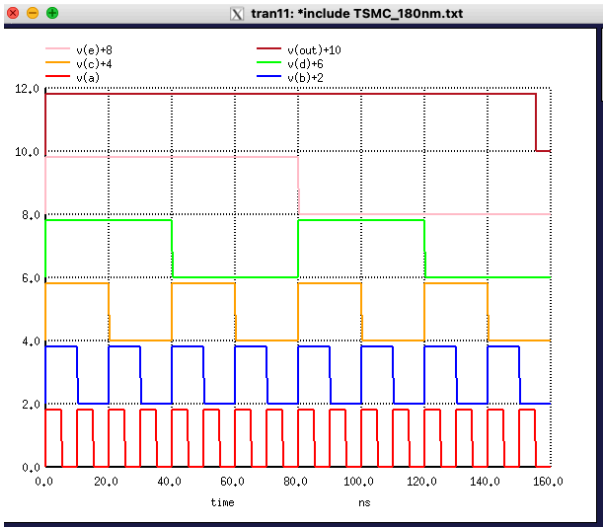


Figure 11 : 5 input or gate functionality

C. *AND gates*

- 2 input

```
** 2 input AND GATE **
.subckt and2 a b out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
   * 2input NAND gate Wn = 2*width_N Wp = width_P*
M1 nout a vdd vdd CMOSP W={width_P}
L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M2 nout b vdd vdd CMOSP W={width_P}
L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}

M3 nout a x gnd CMOSN W={2*width_N}
L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}
M4  x b gnd gnd CMOSN W={2*width_N}
L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}

   *not gate*
M5 out nout vdd vdd CMOSP W={width_P}
L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M6 out nout gnd gnd CMOSN W={width_N}
L={2*LAMBDA}
+ AS={5*width_N*LAMBDA}
PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA}
PD={10*LAMBDA+2*width_N}
.ends and2
```

Figure 12 : 2 input and gate netlist

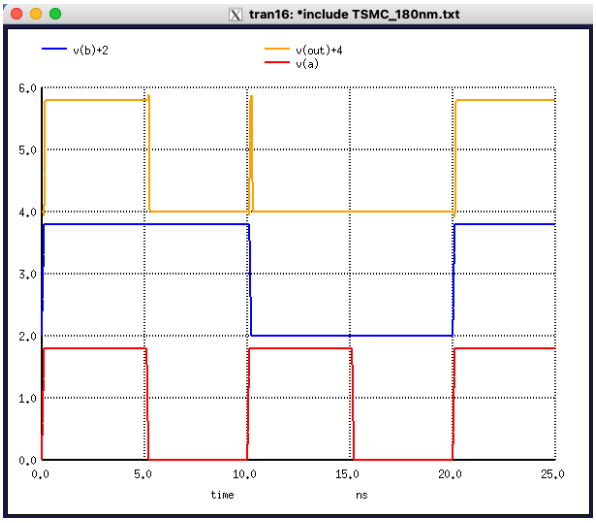


Figure 13 : 2 input and gate functionality

- 3 input

```
** 3 input AND GTE **
.subckt and3 a b c out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
   * 3input NAND gate Wn = 3*width_N Wp = width_P*
M1 nout a vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P} AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M2 nout b vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P} AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M3 nout c vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P} AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}

M4 nout a x gnd CMOSN W={3*width_N} L={2*LAMBDA}
+ AS={5*3*width_N*LAMBDA}
PS={10*LAMBDA+2*3*width_N}
AD={5*3*width_N*LAMBDA}
PD={10*LAMBDA+2*3*width_N}
M5  x b y gnd CMOSN W={3*width_N} L={2*LAMBDA}
+ AS={5*3*width_N*LAMBDA}
PS={10*LAMBDA+2*3*width_N}
AD={5*3*width_N*LAMBDA}
PD={10*LAMBDA+2*3*width_N}
M6 y c gnd gnd CMOSN W={3*width_N} L={2*LAMBDA}
+ AS={5*3*width_N*LAMBDA}
PS={10*LAMBDA+2*3*width_N}
AD={5*3*width_N*LAMBDA}
PD={10*LAMBDA+2*3*width_N}

   *not gate*
M7 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA}
PS={10*LAMBDA+2*width_P} AD={5*width_P*LAMBDA}
PD={10*LAMBDA+2*width_P}
M8 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA}
PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA}
PD={10*LAMBDA+2*width_N}
.ends and3
```

Figure 14 : 3 input and gate netlist

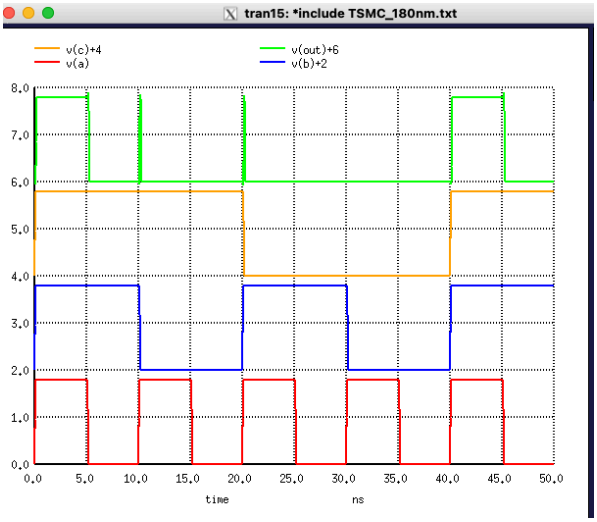


Figure 15 : 3 input and gate functionality

- 4 input

```
.subckt and4 a b c d out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
   * 4input NAND gate Wn = 4*width_N Wp = width_P*
M1 nout a vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M2 nout b vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M3 nout c vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M4 nout d vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}

M5 nout a x gnd CMOSN W={4*width_N} L={2*LAMBDA}
+ AS={5*4*width_N*LAMBDA}
PS={10*LAMBDA+2*4*width_N}
AD={5*4*width_N*LAMBDA}
PD={10*LAMBDA+2*4*width_N}
M6  x b y gnd CMOSN W={4*width_N} L={2*LAMBDA}
+ AS={5*4*width_N*LAMBDA}
PS={10*LAMBDA+2*4*width_N}
AD={5*4*width_N*LAMBDA}
PD={10*LAMBDA+2*4*width_N}
M7 y c z gnd CMOSN W={4*width_N} L={2*LAMBDA}
+ AS={5*4*width_N*LAMBDA}
PS={10*LAMBDA+2*4*width_N}
AD={5*4*width_N*LAMBDA}
PD={10*LAMBDA+2*4*width_N}
M8 z d gnd gnd CMOSN W={4*width_N} L={2*LAMBDA}
+ AS={5*4*width_N*LAMBDA}
PS={10*LAMBDA+2*4*width_N}
AD={5*4*width_N*LAMBDA}
PD={10*LAMBDA+2*4*width_N}

   *not gate*
M9 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M10 out nout gnd gnd CMOSN W={width_N}
L={2*LAMBDA}
+ AS={5*width_N*LAMBDA}
PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA}
PD={10*LAMBDA+2*width_N}
.ends and4
```

Figure 16 : 4 input and gate netlist

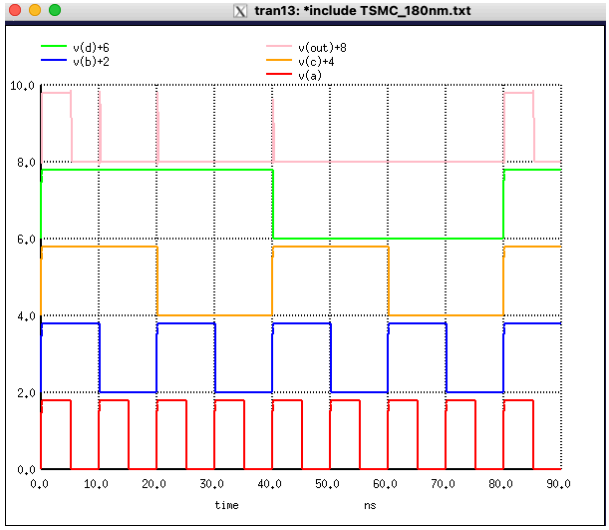


Figure 17 : 4 input and gate functionality

- 5 input

```
.subckt and5 a b c d e out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
   * 5input NAND gate Wn = 5*width_N Wp = width_P*
M1 nout a vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M2 nout b vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M3 nout c vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M4 nout d vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M5 nout e vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}

M6 nout a w gnd CMOSN W={5*width_N} L={2*LAMBDA}
+ AS={5*5*width_N*LAMBDA}
PS={10*LAMBDA+2*5*width_N}
AD={5*5*width_N*LAMBDA}
PD={10*LAMBDA+2*5*width_N}
M7 w b x gnd CMOSN W={5*width_N} L={2*LAMBDA}
+ AS={5*5*width_N*LAMBDA}
PS={10*LAMBDA+2*5*width_N}
AD={5*5*width_N*LAMBDA}
PD={10*LAMBDA+2*5*width_N}
M8 x c y gnd CMOSN W={5*width_N} L={2*LAMBDA}
+ AS={5*5*width_N*LAMBDA}
PS={10*LAMBDA+2*5*width_N}
AD={5*5*width_N*LAMBDA}
PD={10*LAMBDA+2*5*width_N}
M9 y d z gnd CMOSN W={5*width_N} L={2*LAMBDA}
+ AS={5*5*width_N*LAMBDA}
PS={10*LAMBDA+2*5*width_N}
AD={5*5*width_N*LAMBDA}
PD={10*LAMBDA+2*5*width_N}
M10 z e gnd gnd CMOSN W={5*width_N} L={2*LAMBDA}
+ AS={5*5*width_N*LAMBDA}
PS={10*LAMBDA+2*5*width_N}
AD={5*5*width_N*LAMBDA}
PD={10*LAMBDA+2*5*width_N}

   *not gate*
M11 out nout vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M12 out nout gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width  N*LAMBDA} PS={10*LAMBDA+2*width  N}
```

Figure 18 : 5 input and gate netlist

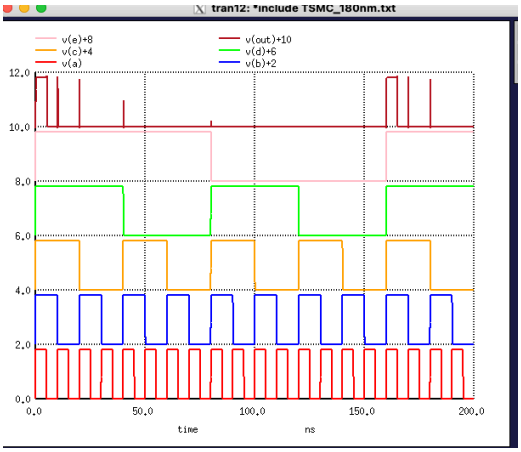


Figure 19 : 5 input and gate functionality

D. *XOR gate*

```
.subckt xor a abar b bbar out vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}

* 2input XOR gate with Wn = 2*width_N and Wp = 2*width_P *
M1 p a vdd vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA}
PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA}
PD={10*LAMBDA+2*2*width_P}
M2 out bbar p vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA}
PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA}
PD={10*LAMBDA+2*2*width_P}
M3 q abar vdd vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA}
PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA}
PD={10*LAMBDA+2*2*width_P}
M4 out b q vdd CMOSP W={2*width_P} L={2*LAMBDA}
+ AS={5*2*width_P*LAMBDA}
PS={10*LAMBDA+2*2*width_P}
AD={5*2*width_P*LAMBDA}
PD={10*LAMBDA+2*2*width_P}

M5 out a r gnd CMOSN W={2*width_N} L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}
M6 r b gnd gnd CMOSN W={2*width_N} L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}
M7 out abar s gnd CMOSN W={2*width_N} L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}
M8 s bbar gnd gnd CMOSN W={2*width_N} L={2*LAMBDA}
+ AS={5*2*width_N*LAMBDA}
PS={10*LAMBDA+2*2*width_N}
AD={5*2*width_N*LAMBDA}
PD={10*LAMBDA+2*2*width_N}
.ends xor
```

Figure 20 : 2 input xor gate netlist

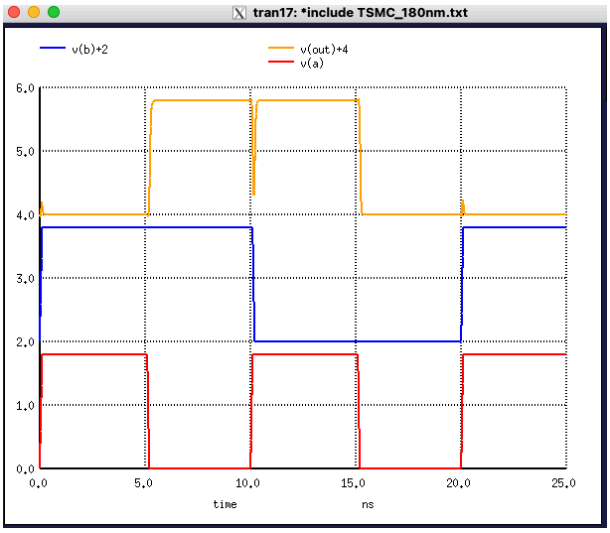


Figure 21 : 2 input xor gate functionality

E. *D flip-flop*

```
.subckt ffposedge d q clk vdd gnd
.param width_P = {20*LAMBDA}
.param width_N = {10*LAMBDA}
M1 x d gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 x clk a vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M3 a d vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}

M4 b clk gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M5 y x b gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M6 y clk vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}

M7 c y gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M8 qbar clk c gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M9 qbar y vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}

M10 q qbar vdd vdd CMOSP W={width_P} L={2*LAMBDA}
+ AS={5*width_P*LAMBDA} PS={10*LAMBDA+2*width_P}
AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
M11 q qbar gnd gnd CMOSN W={width_N} L={2*LAMBDA}
+ AS={5*width_N*LAMBDA} PS={10*LAMBDA+2*width_N}
AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends ffposedge
```

Figure 22 : TSPC flip-flop netlist

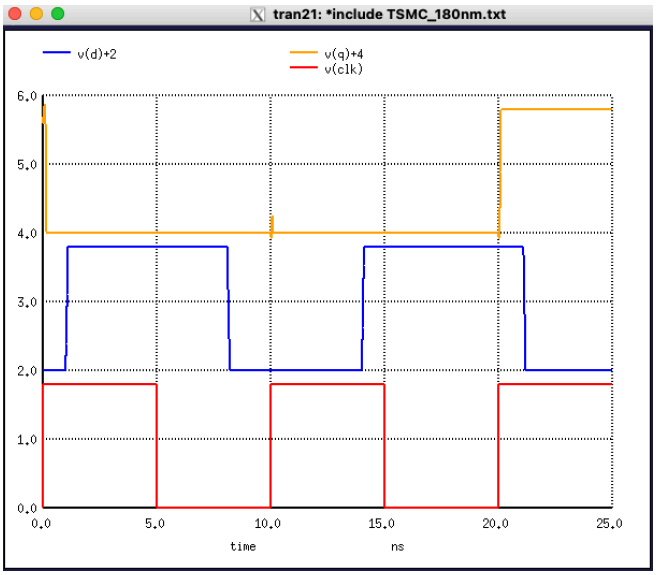


Figure 23 : TSPC flip-flop functionality

V. D Flip-Flop Timing Analysis

A. *SETUP time*

When both clock,clk, and input D signal change at same time , output does not follow input.

Figure 24 : TSPC flip-flop setup time analysis-I

Then by trial-and-error method , checking values for setup time, by changing D input just before rising edge of clock. Setup time was found to be ***0.12ns.***

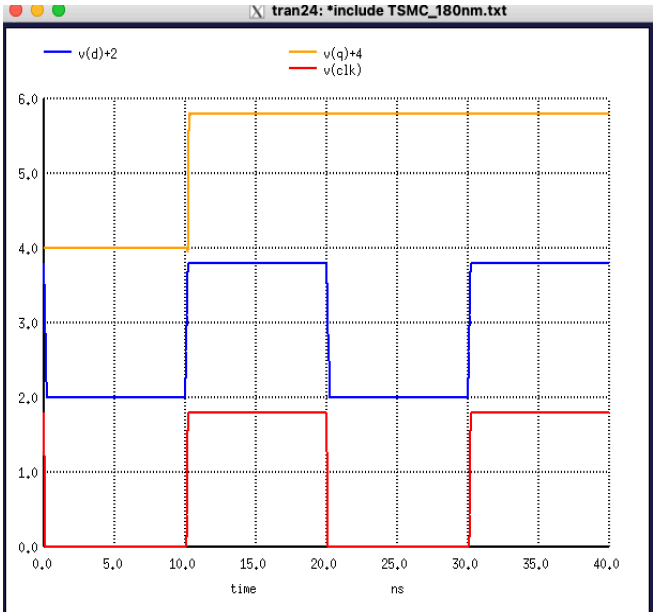


Figure 25 : TSPC flip-flop setup time analysis-II

B. *HOLD time*

We know for TSPC flip-flop , hold time is 0 second.

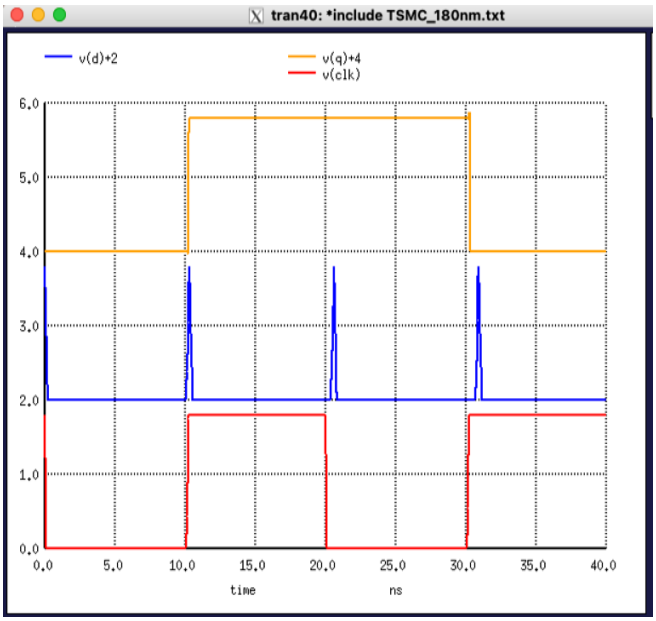


Figure 26 : TSPC flip-flop hold time analysis

C. *Clock to Q delay*

Using meas in NGSpice , I computed clock to Q delay , which came out to be ***85ps*** .

```
.control
run
meas tran tcq trig v(clk) val=0.9 rise=1 targ v(Q) val=0.9 rise=1
plot v(clk) v(d)+2 v(q)+4
set hcopypscolor = 1
set color0=white
set color1=black
.endc
```

Figure 27 : NGSpice code for tcq delay

```
No. of Data Rows : 135
tcq            = 8.525702e-11 targ= 1.023526e-08 trig= 1.015000e-08
```

Figure 28 : TSPC flip-flop tcq time

- $T_{SU} = 0.12ns$
- $T_H = 0s$
- $T_{CQ} = 0.085ns$

VI. Stick Diagram

Legends :

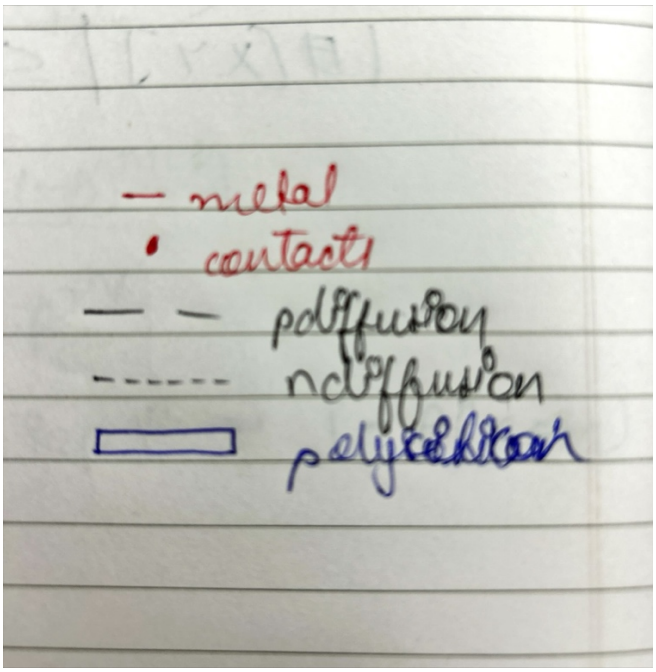


Figure 28 : Legends

- NOT gate

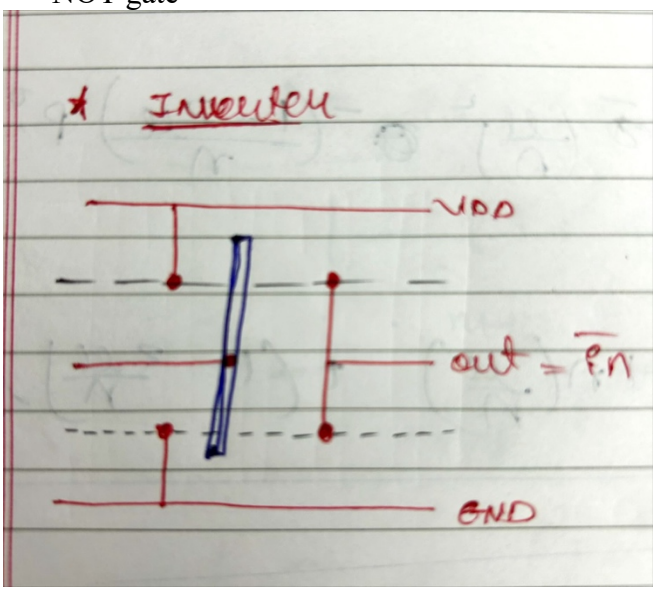


Figure 29 : stick diagram of not gate

- NOR gates

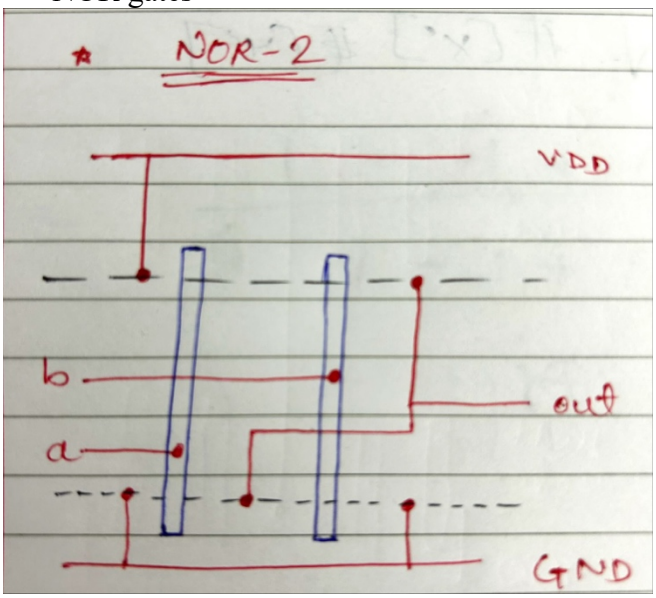


Figure 30 : stick diagram of 2 input nor gate

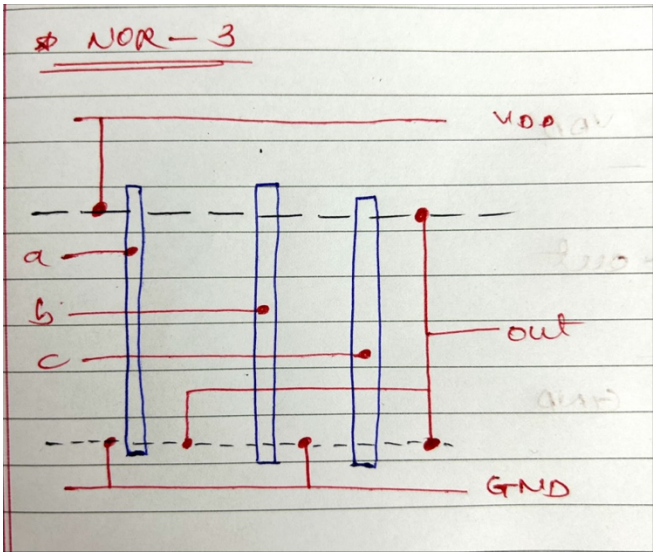


Figure 31 : stick diagram of 3 input nor gate

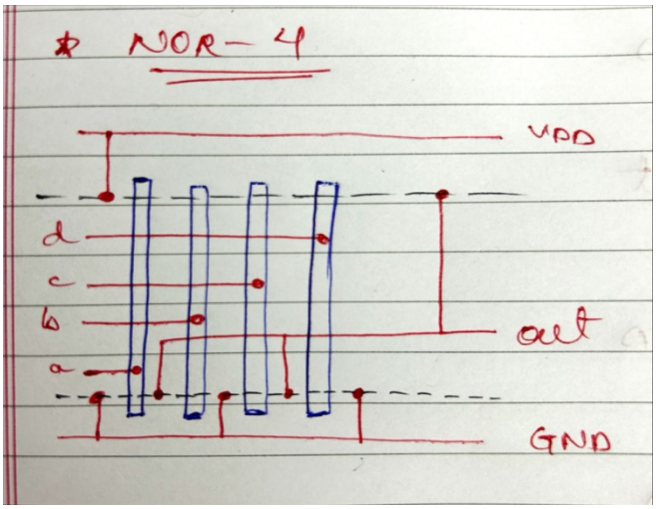


Figure 32 : stick diagram of 4 input nor gate

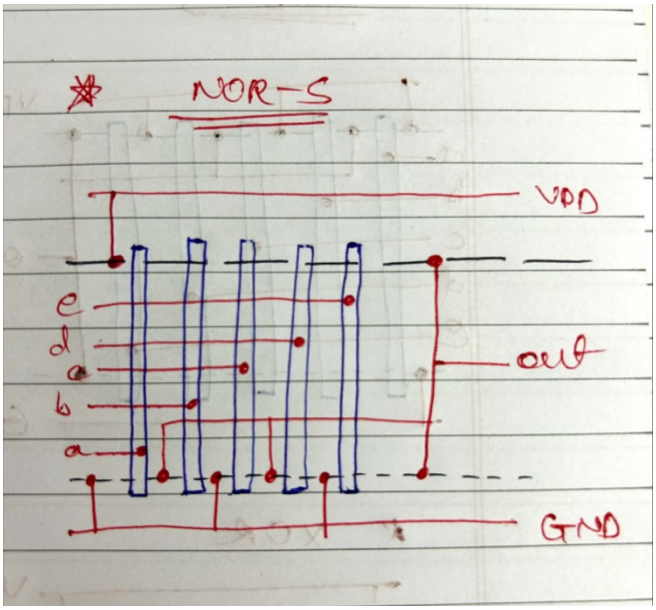


Figure 33 : stick diagram of 5 input nor gate

- NAND gates

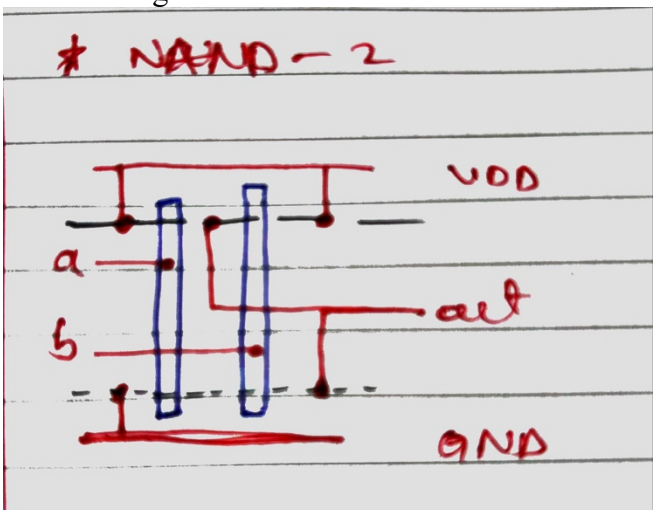


Figure 34 : stick diagram of 2 input nand gate

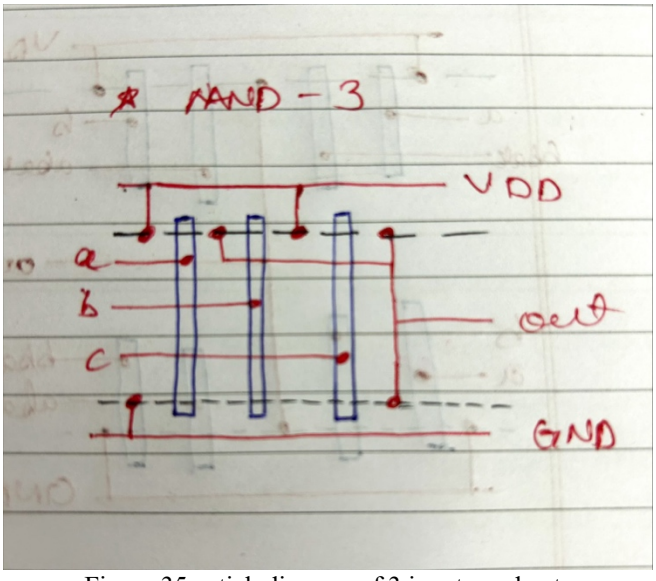


Figure 35 : stick diagram of 3 input nand gate

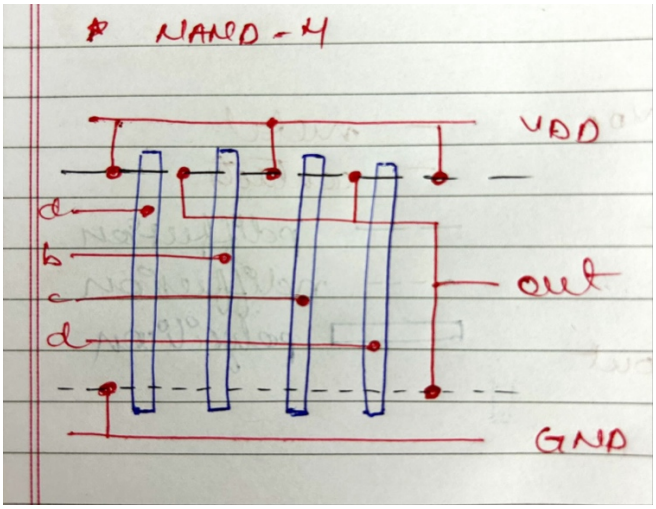


Figure 36 : stick diagram of 4 input nand gate

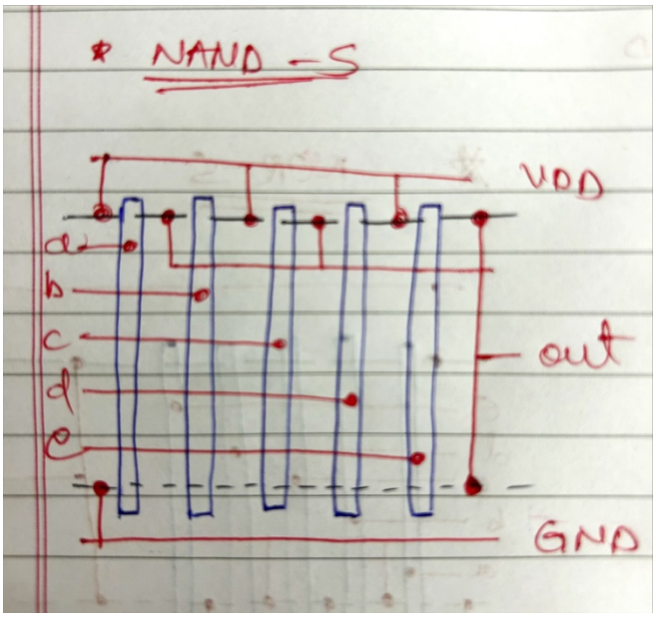


Figure 37 : stick diagram of 5 input nand gate

- XOR gate

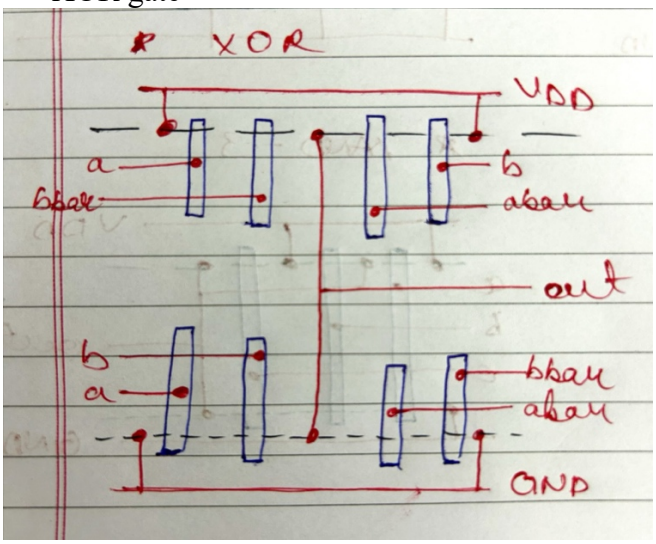


Figure 38 : stick diagram of 2 input xor gate

VII. MAGIC LAYOUT & POST-LAYOUT SIMULATION

A. *NOT gate*

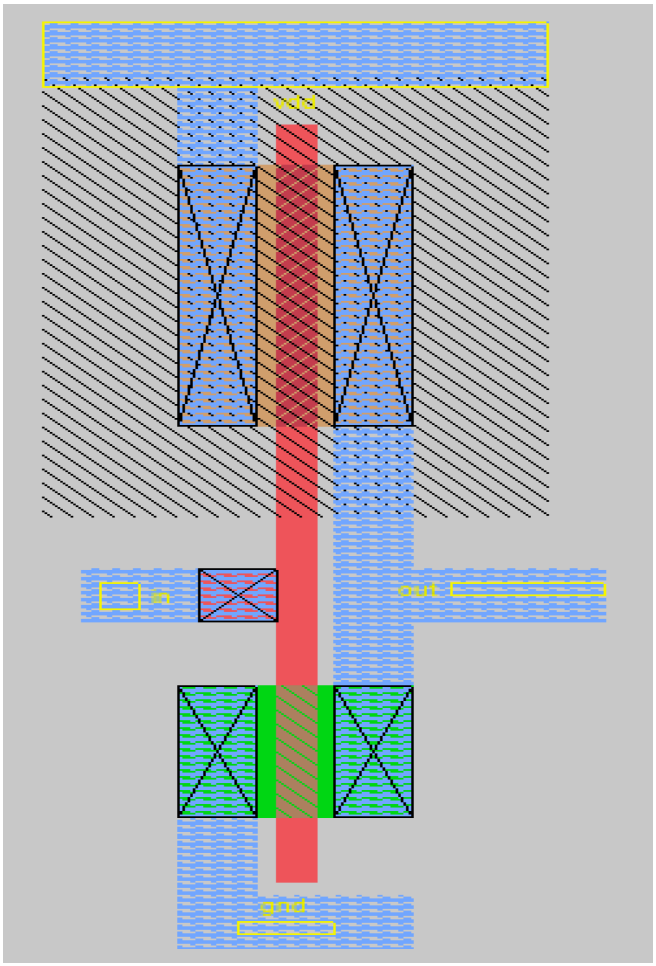


Figure 39 : Not gate layout

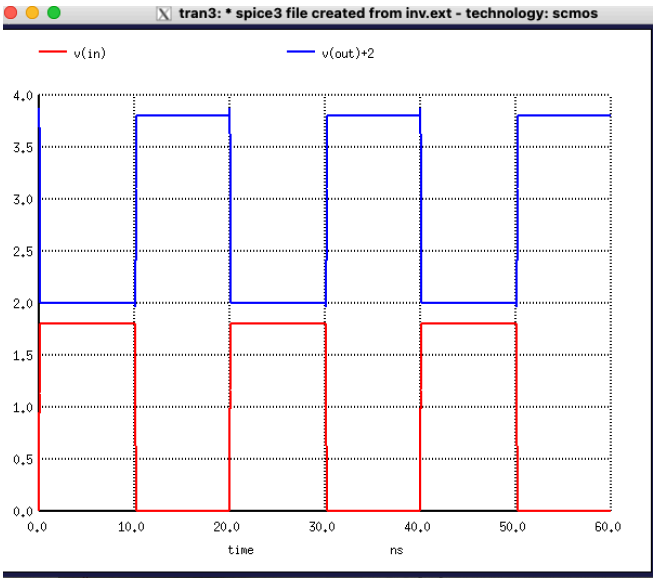


Figure 40 : Not gate functionality

B. *OR gates*

- 2 input

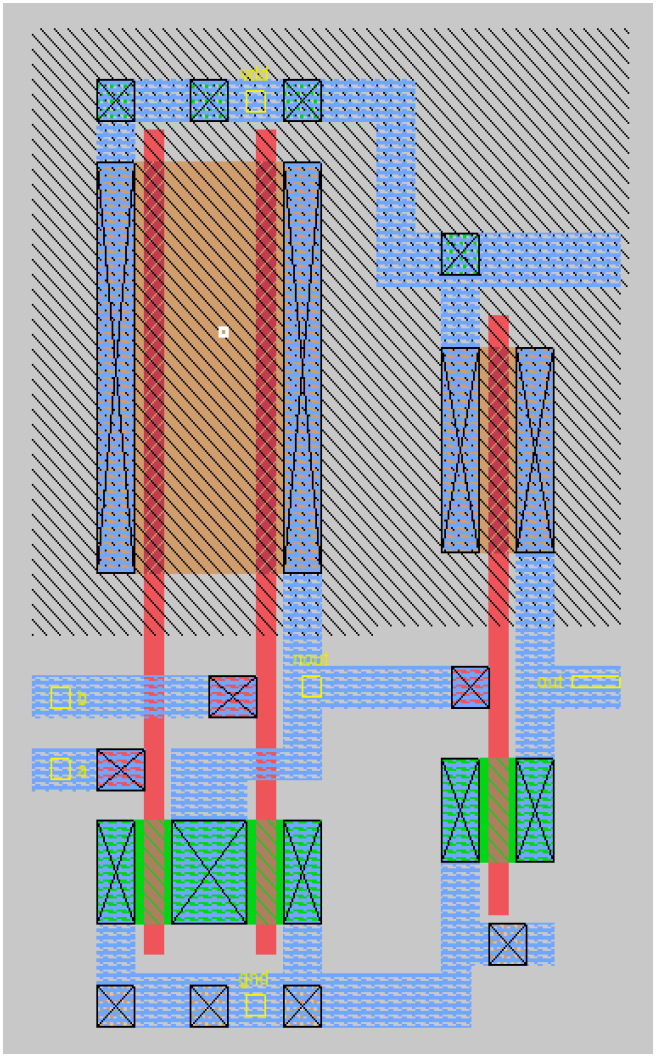


Figure 41 : 2 input or gate layout

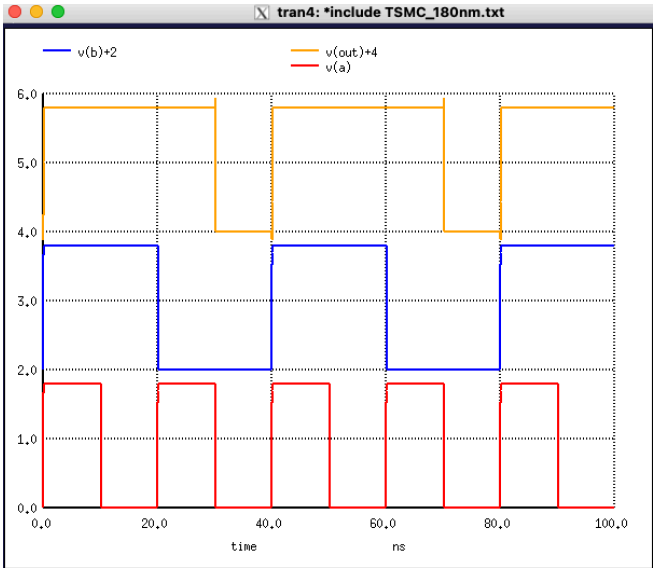


Figure 42 : 2 input or gate functionality

- 3 input

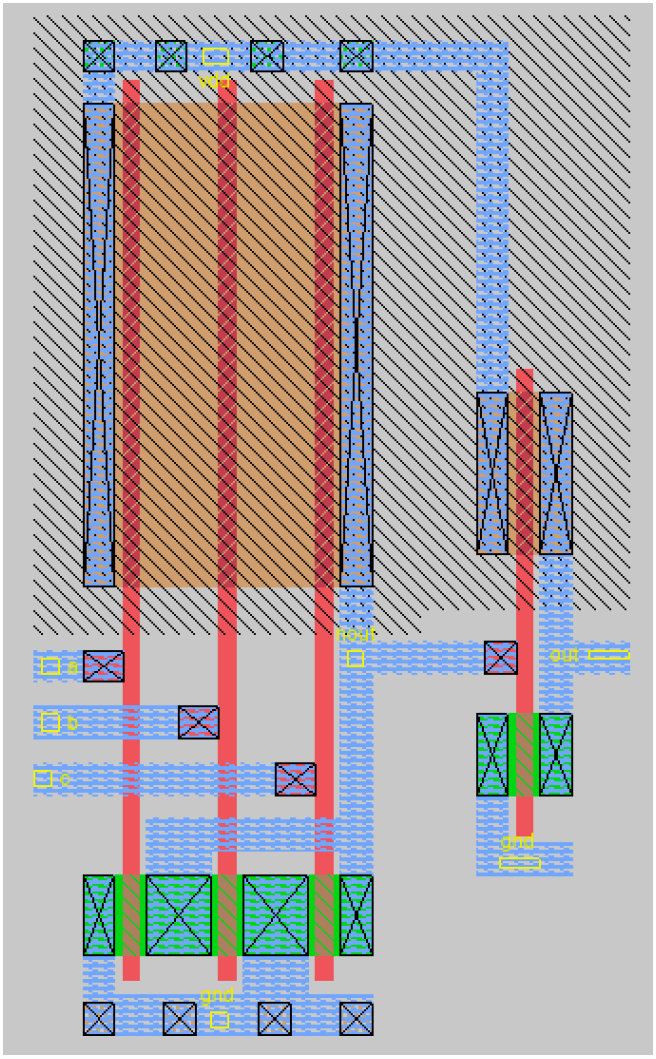


Figure 43 : 3 input or gate layout

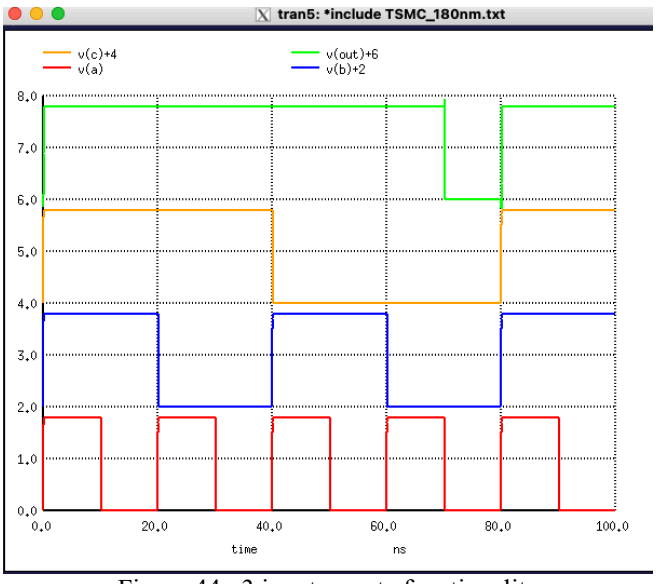


Figure 44 : 3 input or gate functionality

- 4 input

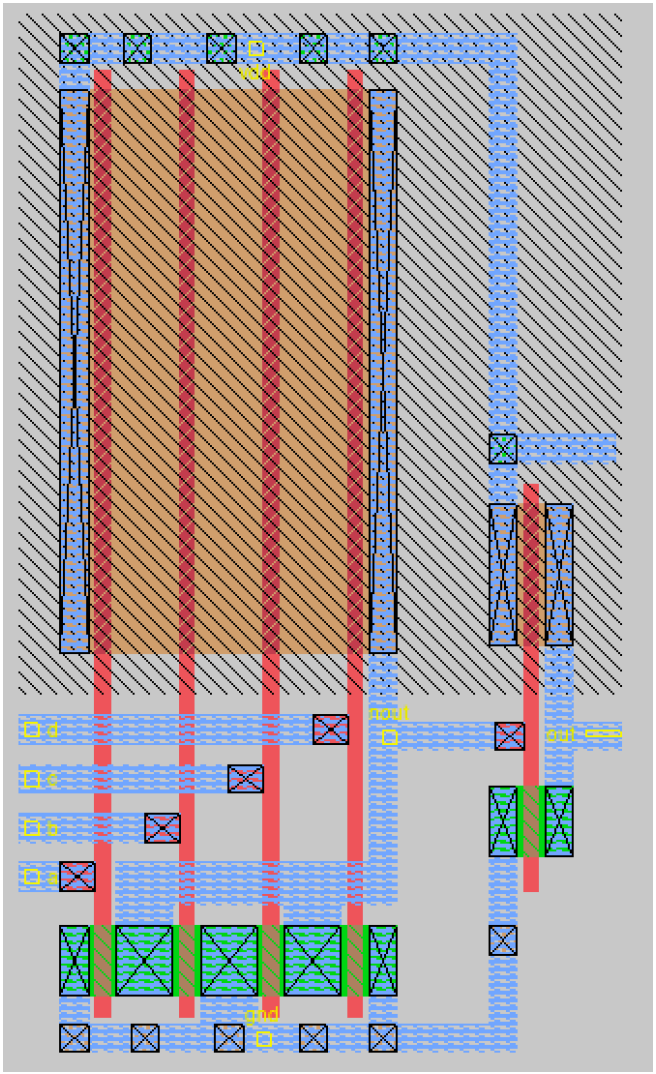


Figure 45 : 4 input or gate layout

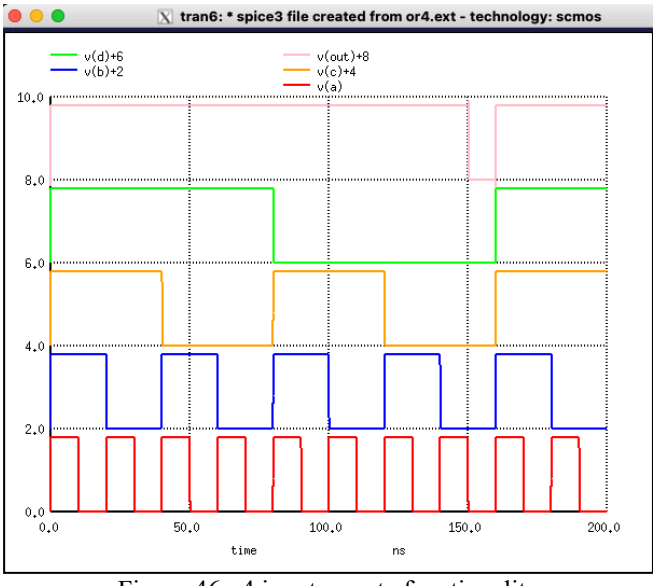


Figure 46 : 4 input or gate functionality

- 5 input

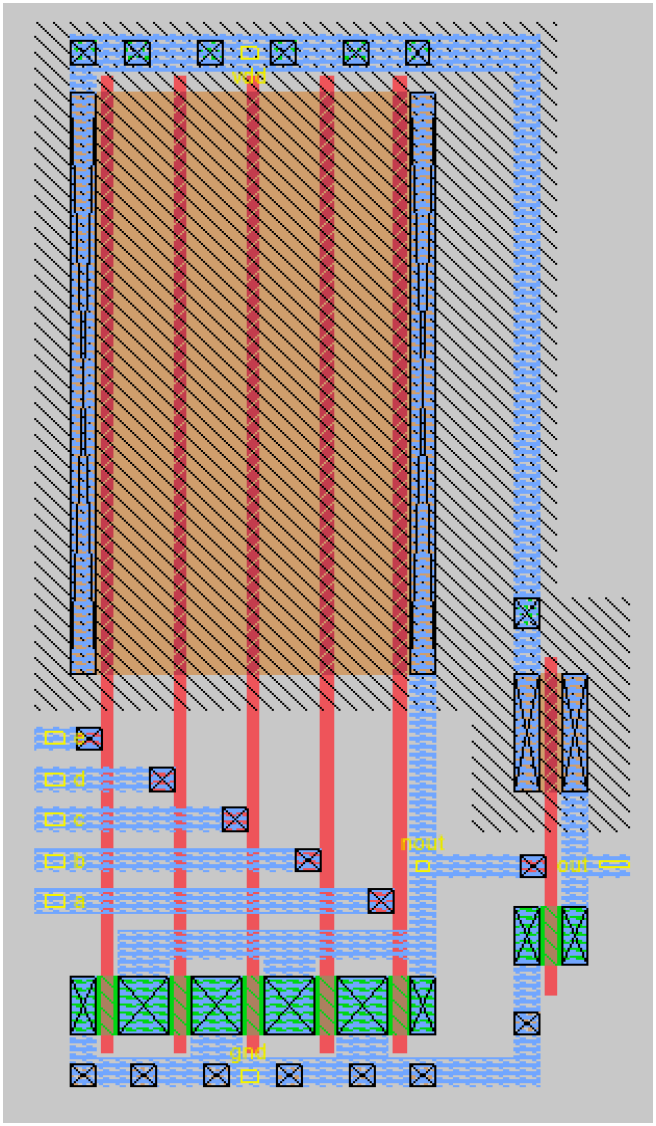


Figure 47 : 5 input or gate layout

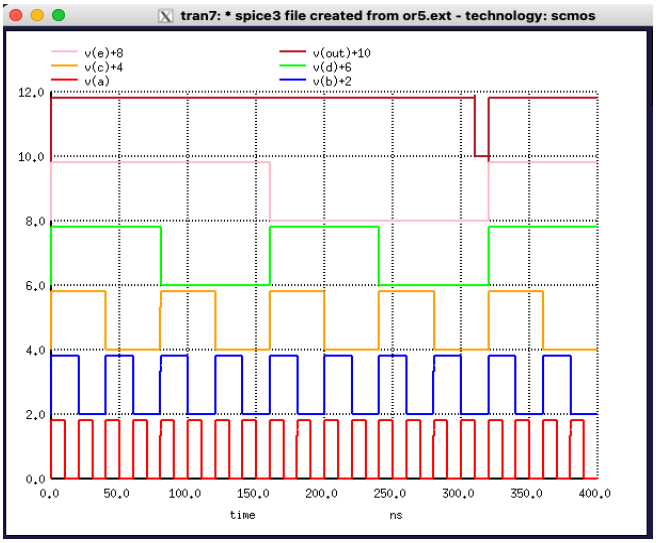


Figure 48 : 5 input or gate functionality

C. AND gates

- 2 input

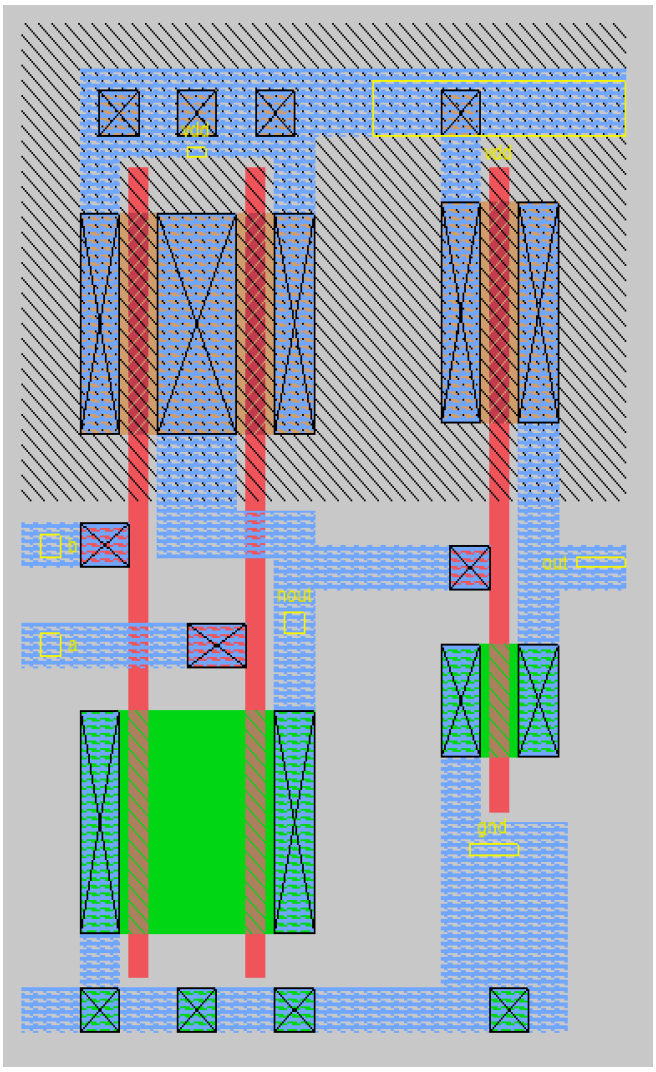


Figure 49 : 2 input and gate layout

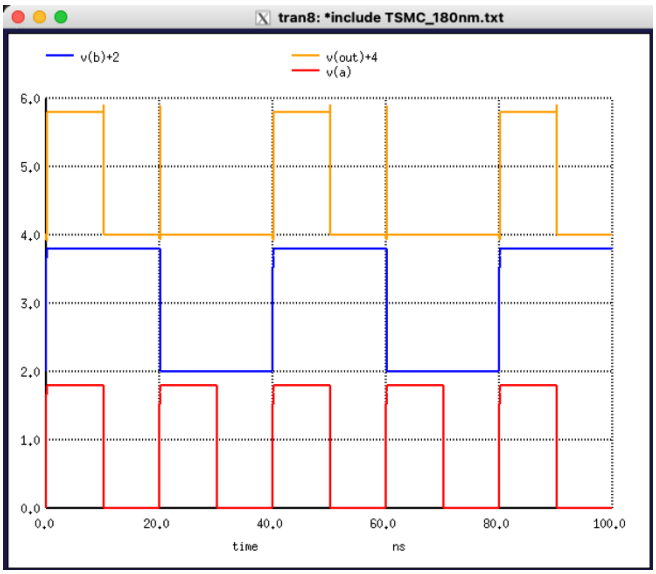


Figure 50 : 2 input and gate functionality

- 3 input

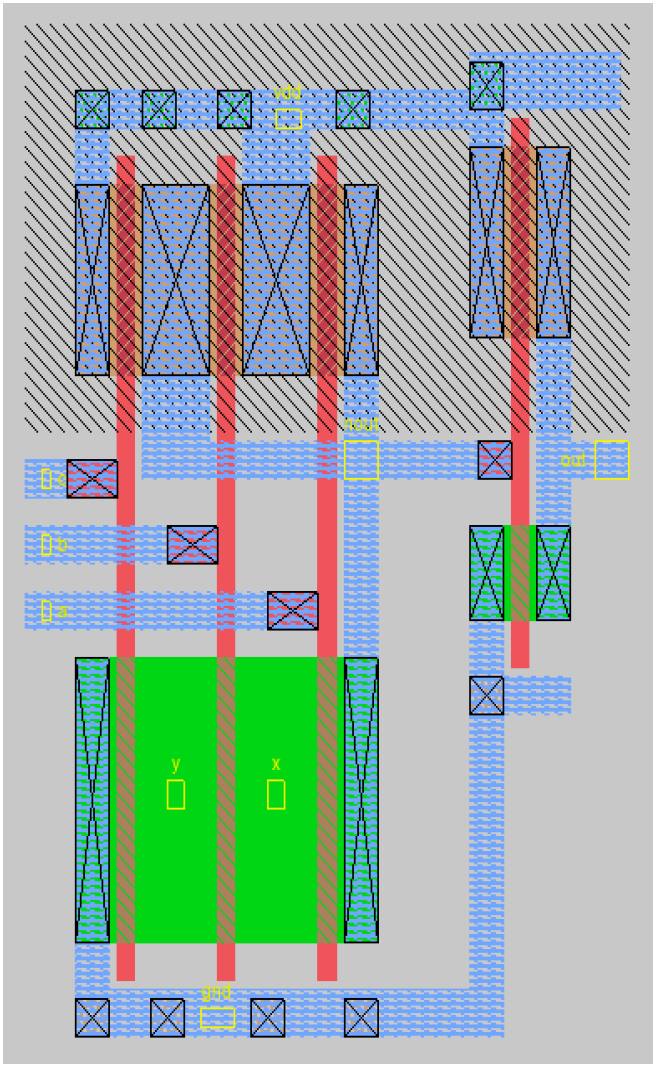


Figure 51 : 3 input and gate layout

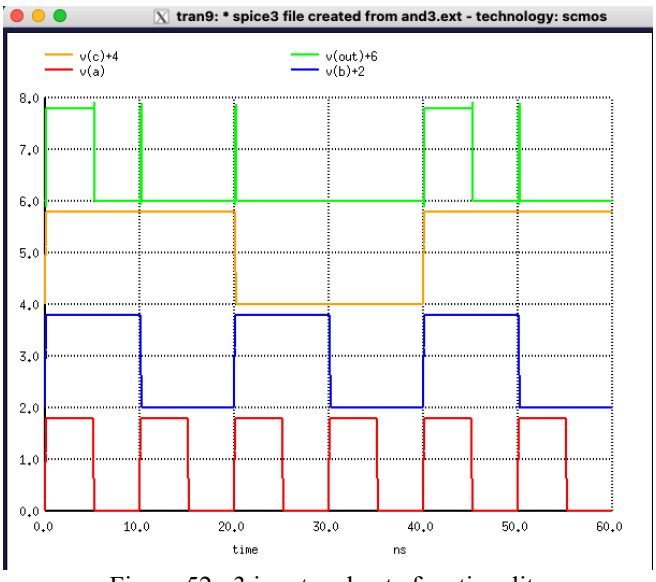


Figure 52 : 3 input and gate functionality

- 4 input

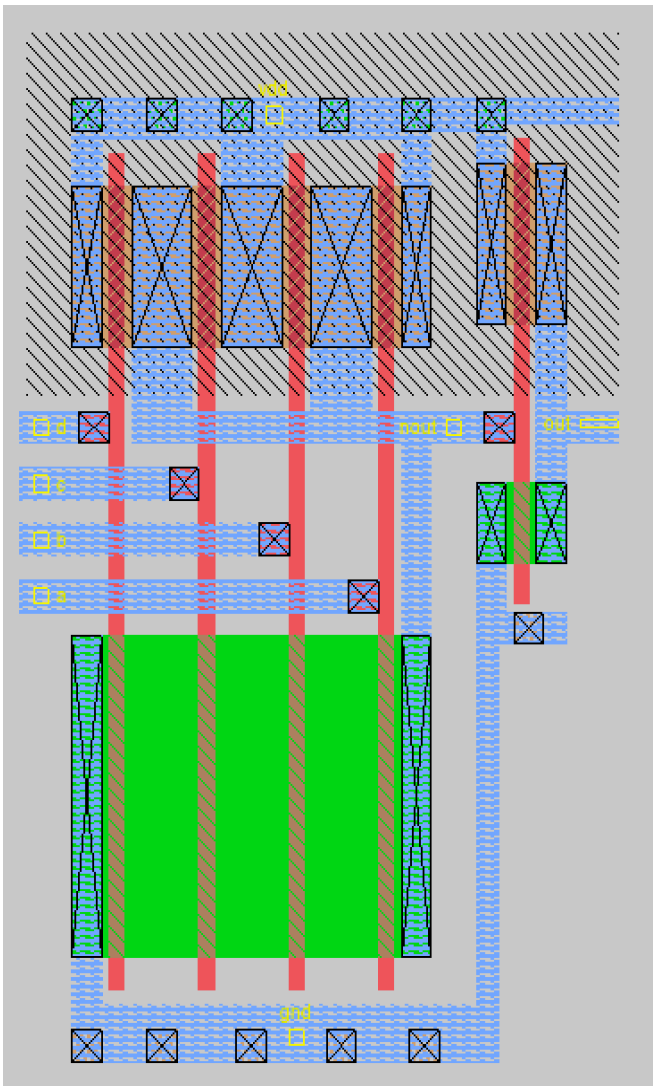


Figure 53 : 4input and gate functionality

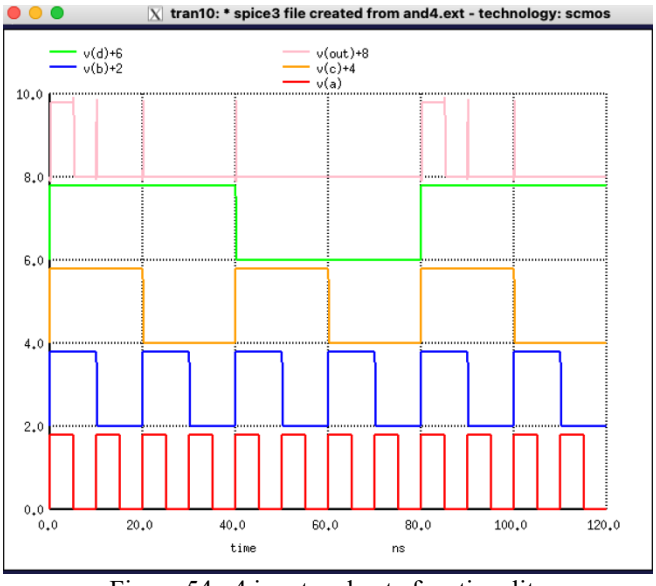


Figure 54 : 4 input and gate functionality

- 5 input

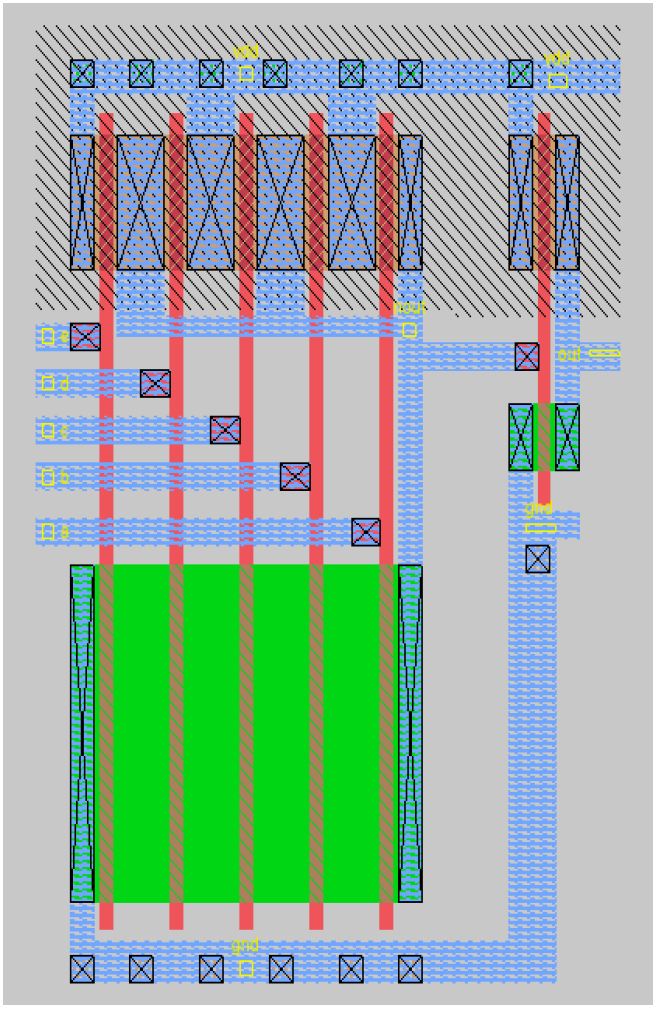


Figure 55 : 5input and gate functionality

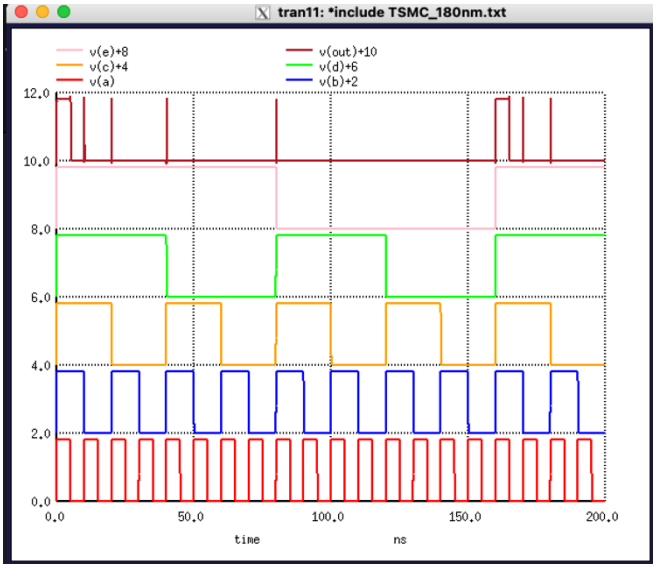


Figure 56 : 5 input and gate functionality

D. *XOR gate*

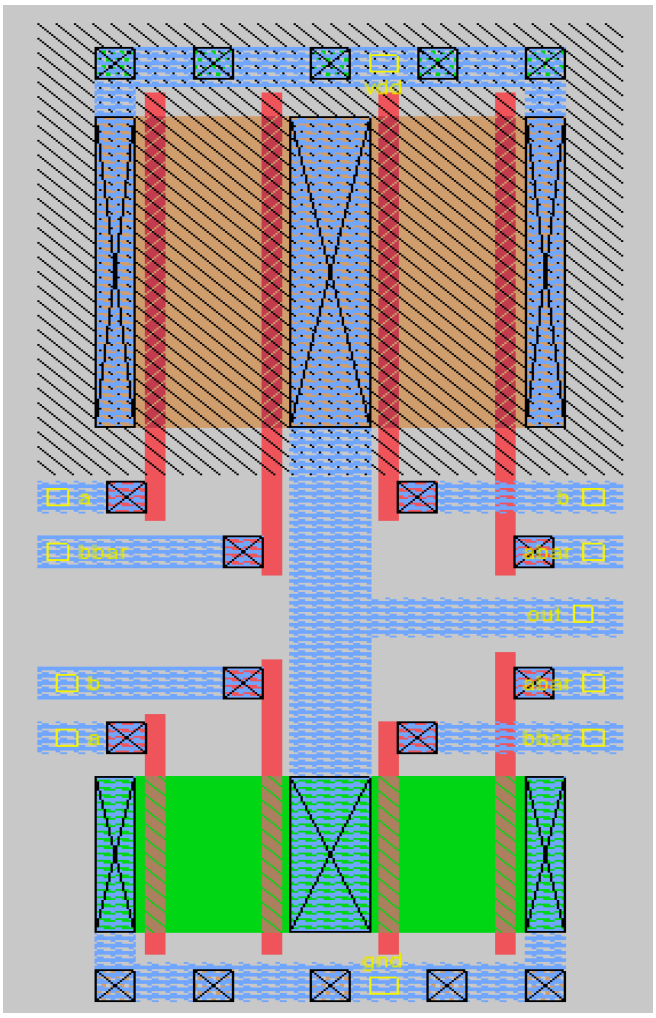


Figure 57 : 2 input xor layout

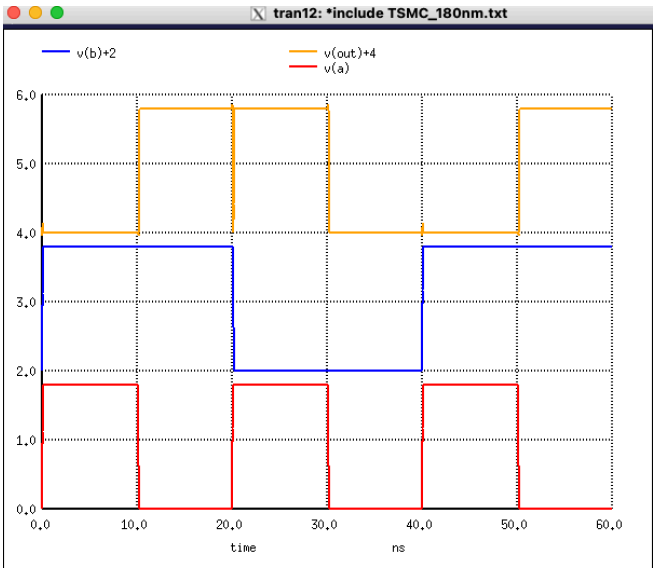


Figure 58 : 2 input and xor gate functionality

E. D flip-flop

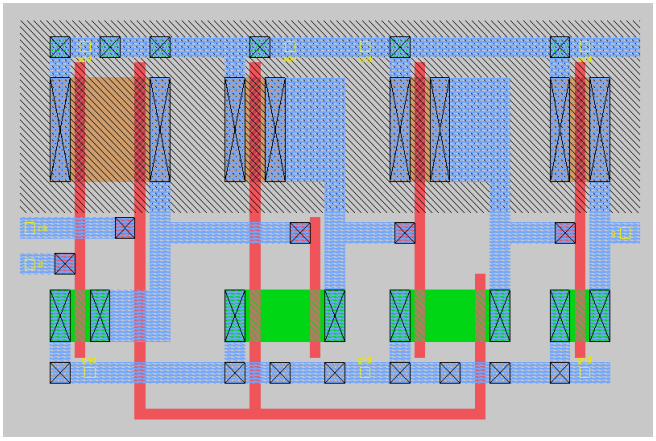


Figure 59 : D flip-flop layout

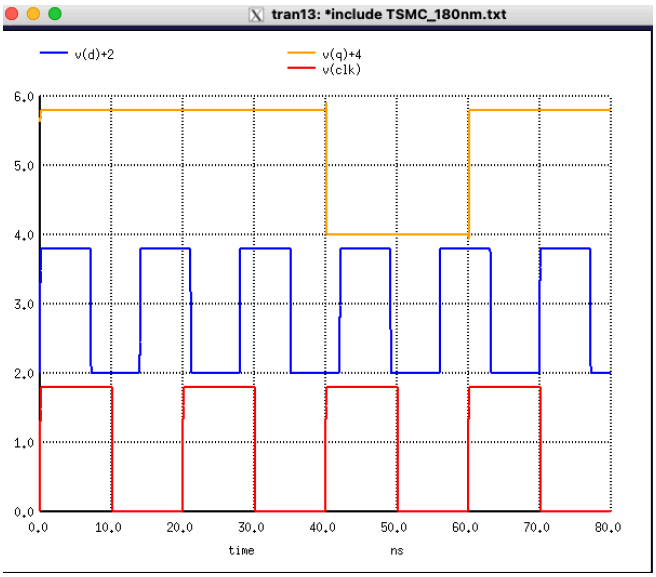


Figure 60 : D flip-flop functionality

VIII. INTEGRATION OF COMPLETE CIRCUIT – PRE-LAYOUT

Complete circuit has been connected according to figure 61 . To check functionality of circuit, I have demonstrated some examples below :

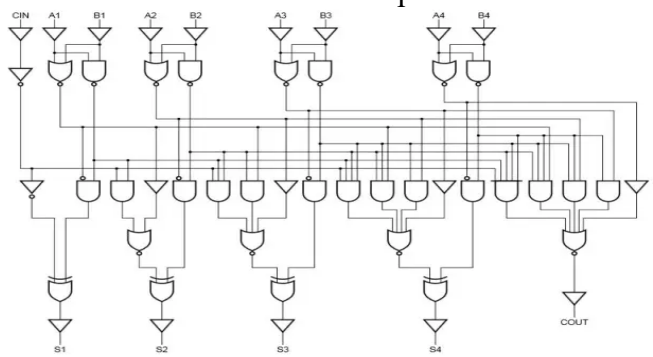


Figure 61 : Complete CLA circuit

Example 1 : c0 = 1

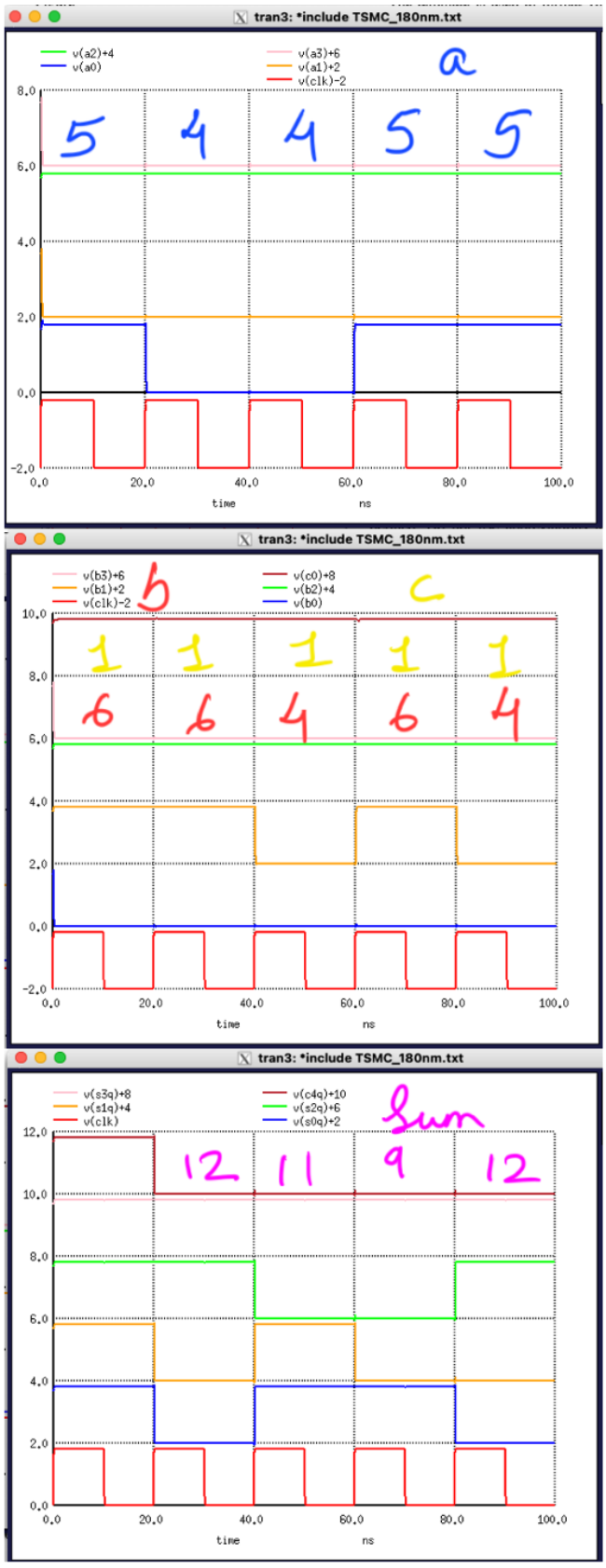


Figure 62 : example 1

Example 2 : c0 = 0

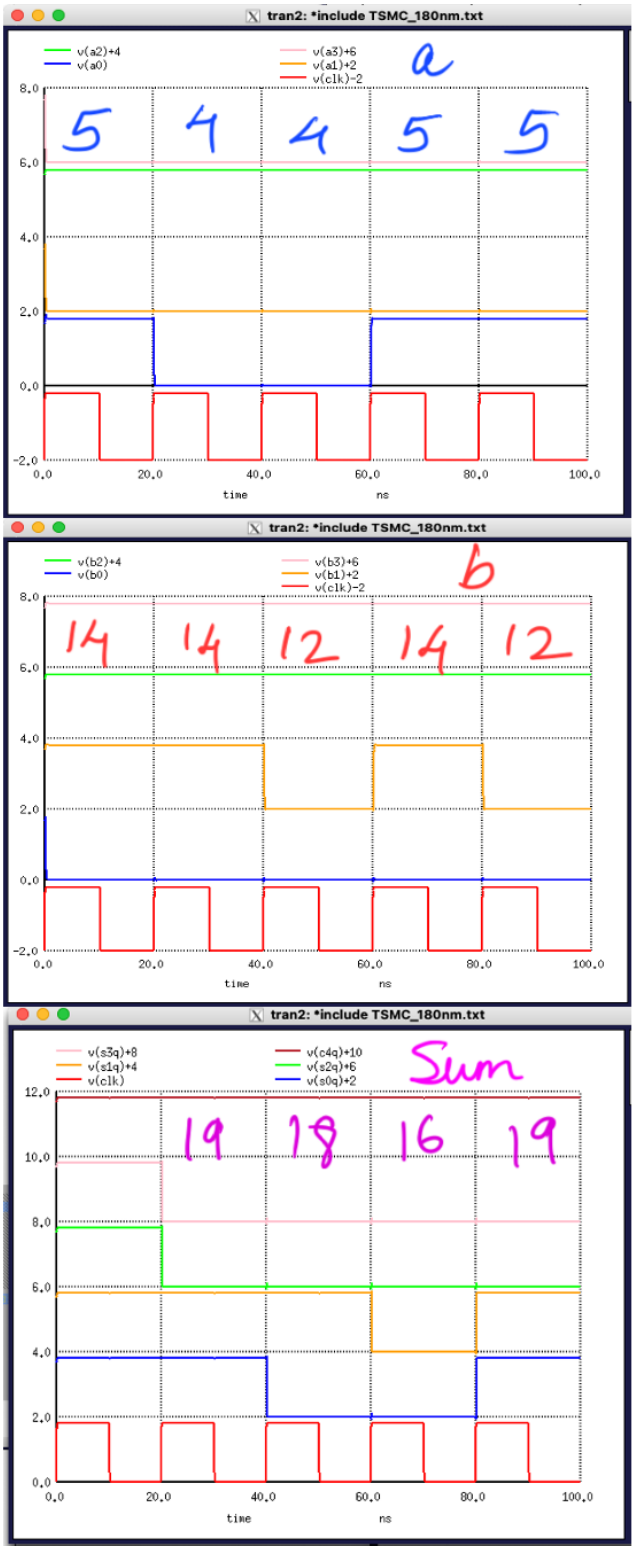
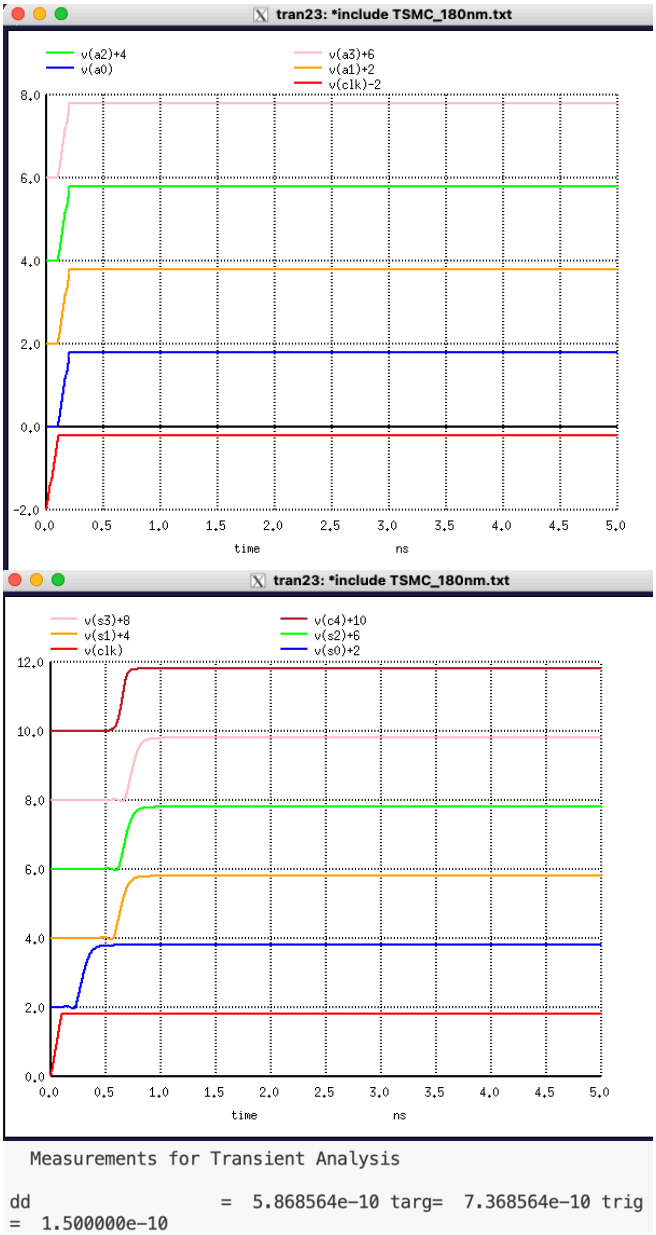


Figure 62 : example 2

A. *Worst Case Delay*

When input is a = 1111 (15) , b = 1111 (15) and c0 = 1, we get output as s = 1111(15) and c4 = 1. This means input a= 15 + b= 15 + c0=1 ,results into 31.
Max Rise Time delay is 0.58ns

```
.meas tran dd trig v(a0) val=0.9 rise=1
+targ v(s3) val=0.9 rise=1
.tran  0.01n 5ns
.control
```



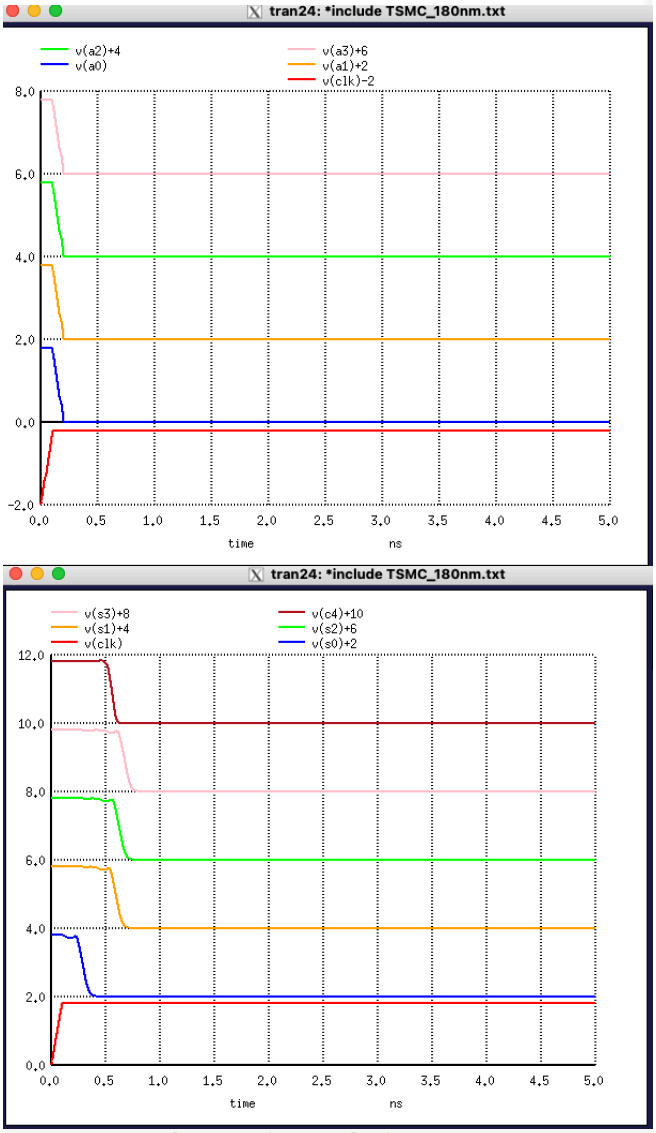

```
  Measurements for Transient Analysis

dd                = 5.868564e-10 targ=  7.368564e-10 trig
=  1.500000e-10
```

Figure 63 : Max rise time

When input is a = 0000 (0) , b = 0 (0) and c0 = 0, we get output as s = 0000(0) and c4 = 0. This means input a= 0 + b=0 +c=0 results into zero.
Max Fall Time delay is 0.52ns

```
.meas tran falldelay trig v(a0) val=0.9 fall=1
+targ v(s3) val=0.9 fall=1
.tran  0.01n 5ns
.control
run
```




```
  Measurements for Transient Analysis

falldelay            =  5.198794e-10 targ=  6.698794e-10 trig
=  1.500000e-10
```

Figure 64 : Max fall time

Hence max delay is 0.58ns.

T_{PD} = 0.58ns

Maximum clock speed is given as $T_{CLK} = T_{PDMAX} + T_{SU} + T_{CQ}$. Therefore $T_{CLK} = 0.785$ns.

IX. Complete Floor Plan on Magic

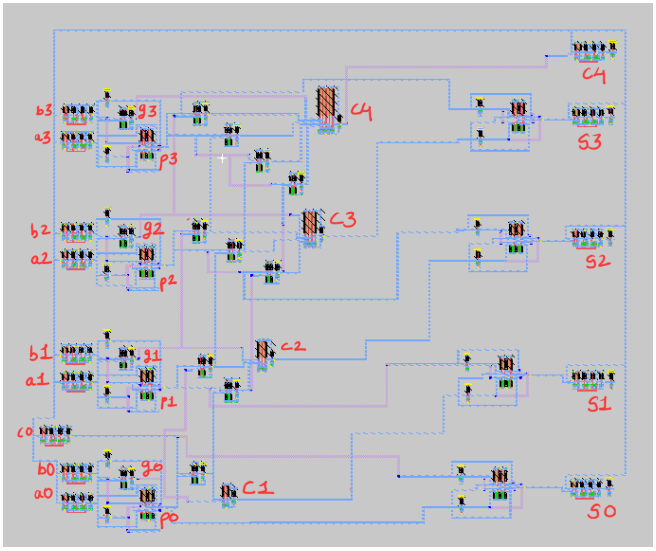


Figure 65 : Complete Magic Layoyt

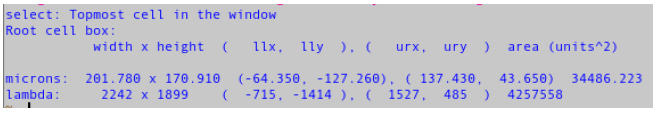


Figure 66 : Area and dimensions

Width = 201.78 µm , Length = 170.91 µm

Area = 34486.223 μm^2

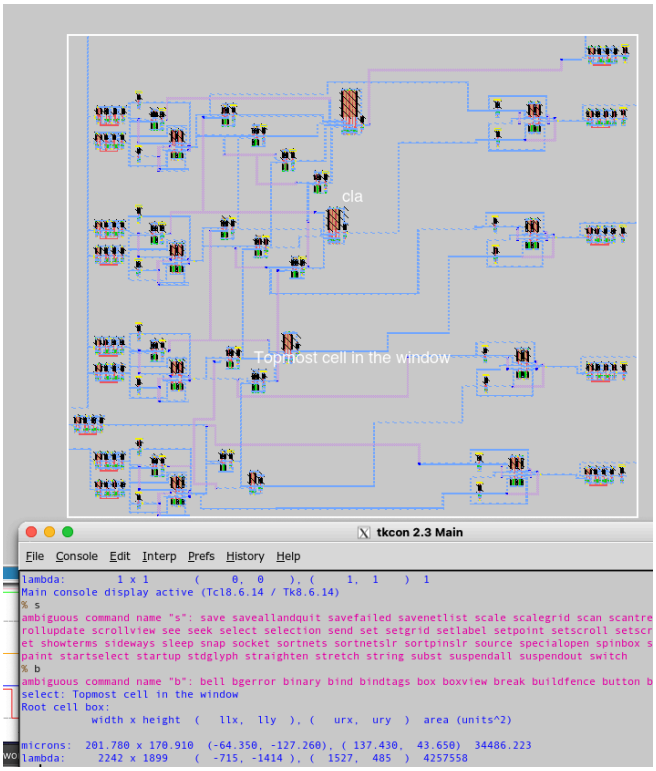


Figure 67: Combined of figure 66 and 65

X. Post-Layout Simulation

A. *Timing Analysis for D flip-flop*

- Setup time:

When input and output are changed at same time.

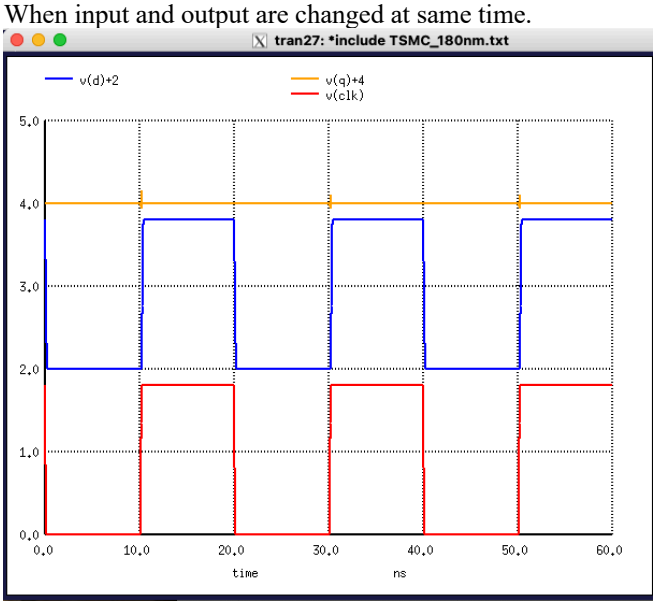


Figure 68 : Setup time analysis-I

By trial and error , ***T_{SU} = 0.13ns***

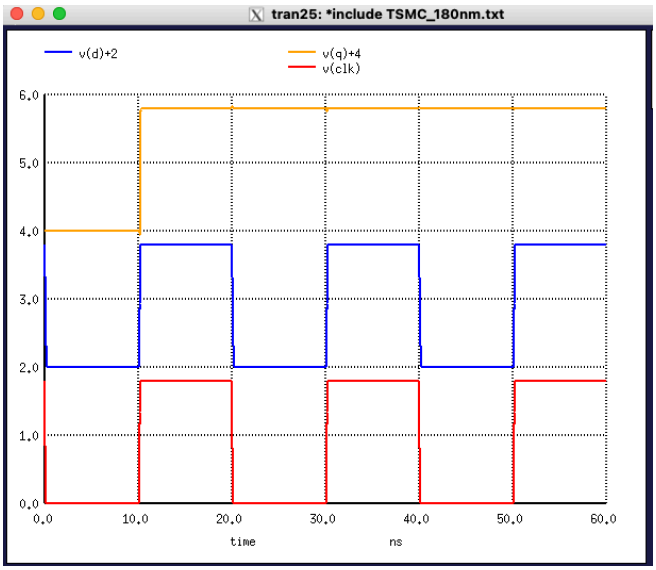


Figure 69 : Setup time analysis-II

- Hold time :

  We know hold time for TSPC flip-flop is 0 seconds.

- Clock to Q delay

  Using NGSpice command

```
run
meas tran tcq trig v(clk) val=0.9 rise=1 targ v(q) val=0.9 rise=1
plot v(clk) v(d)+2 v(q)+4
set hcopypscolor = 1
set color0=white
set color1=black
```

Figure 70 : Tcq NGSpice command

T_{CQ} = 0.075ns

```
Reference value :  1.84750e-08
No. of Data Rows : 6068
tcq                =  7.542159e-11 targ=  1.022542e-08 trig=  1.015000e-08
```

Figure 71 : Tcq NGSpice terminal

- ***T_{SU} = 0.13ns***
- ***T_H = 0s***
- ***T_{CQ} =0.075ns***

B. *CLA adder delay*

Same as previous method I found max fall and max rise time. Maximum fall delay is 0.37ns , while maximum rise delay is 0.56ns. Hence Tpd = 0.56ns.

```
Measurements for Transient Analysis
risedelay          =  5.629839e-10 targ=  5.829839e-10 trig=  2.000000e-11
```

Figure 72 : Rise time delay

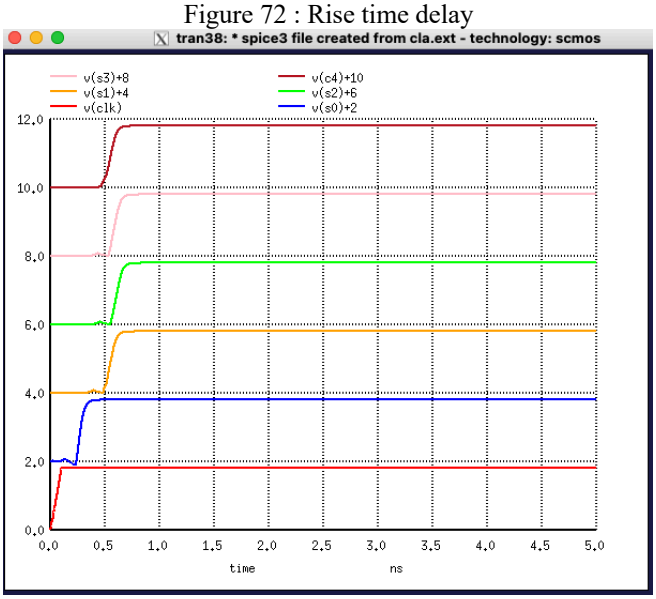


Figure 73 : Rise time delay graph -I

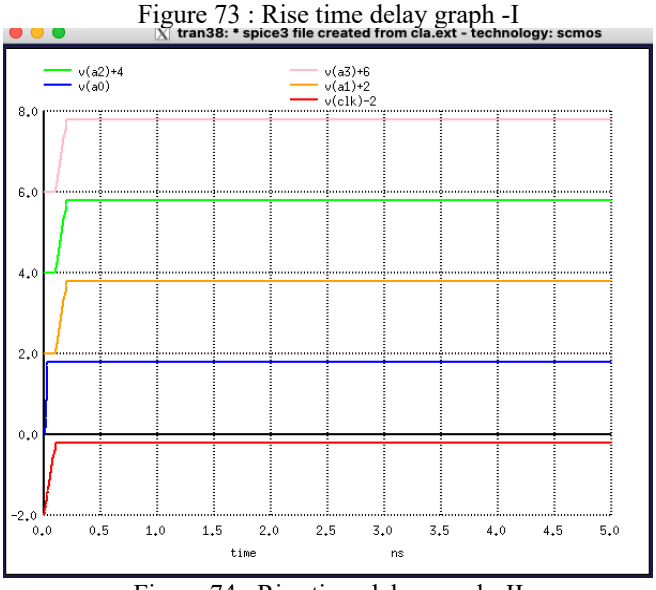


Figure 74 : Rise time delay graph -II

```
Measurements for Transient Analysis

falldelay          =  3.787571e-10 targ=  5.287571e-10 trig=  1.500000e-10
```

Figure 75 : Fall time

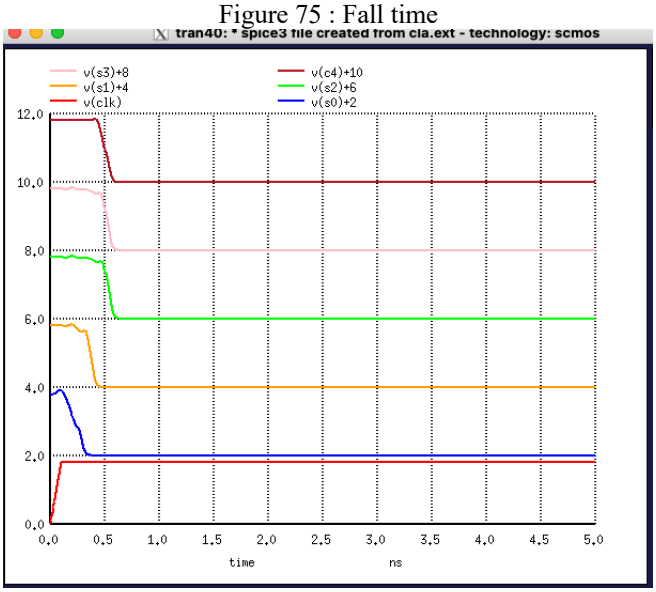


Figure 76 : Fall time delay graph -I

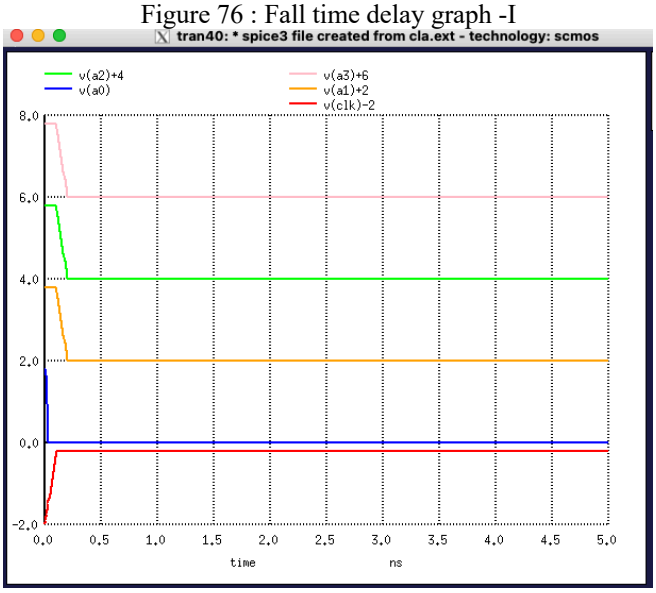


Figure 77 : Fall time delay graph -II

Table1 :Pre vs Post layout for D flip-flop

	Setup Time	Clk to Q delay
Pre-layout	0.12n	0.085n
Post-layout	0.13n	0.075n

Table 2: Pre vs Post layout for CLA adder

	Max Rise Time	Max Fall Time
Pre-layout	0.58ns	0.52ns
Post-layout	0.56ns	0.37ns

Hence , $\underline{\boldsymbol{T_{CLK} = T_{SU} + T_{COMAX} + T_{PDMAX} = 0.765ns}}$. Surprisingly , it is less then pre-layout T_{CLK} of 0.785ns.

Table 3: Pre vs Post layout clock timing

	Minimum clock time
Pre-layout	0.785ns
Post-layout	0.765ns

Table 4: Pre vs Post layout Maximum Clock Frequency

	Maximum clock frequency
Pre-layout	1.27GHz
Post-layout	1.30GHz

XI. Verilog HDL Result

Figure 78 : Output of Verilog

```
dhimantbhuva@Dhimants-MacBook-Air Verilog % iverilog -o example cla_tb.v cla.v
dhimantbhuva@Dhimants-MacBook-Air Verilog % vvp example
VCD info: dumpfile output.vcd opened for output.
Time: 5000 | a: 0000 | b: 0000 | c0: 1 | s: xxxx | c4: x
Time: 15000 | a: 0001 | b: 1000 | c0: 0 | s: xxxx | c4: x
Time: 25000 | a: 0110 | b: 0101 | c0: 1 | s: 0001 | c4: 0
Time: 35000 | a: 1111 | b: 1111 | c0: 0 | s: 1001 | c4: 0
Time: 45000 | a: 1010 | b: 0101 | c0: 1 | s: 1100 | c4: 0
Time: 55000 | a: 0011 | b: 1111 | c0: 1 | s: 1110 | c4: 1
Time: 65000 | a: 0011 | b: 1111 | c0: 1 | s: 0000 | c4: 1
cla_tb.v:34: $finish called at 68000 (1ps)
```

Figure 79 : Output of Verilog on terminal

XII. FPGA Implementation

- Example 1 : a = 9 (red), b = 4 (purple), c = 0 (yellow). Output, s = 1101 (green LEDs) meaning 13. Leftmost bit is most significant bit.

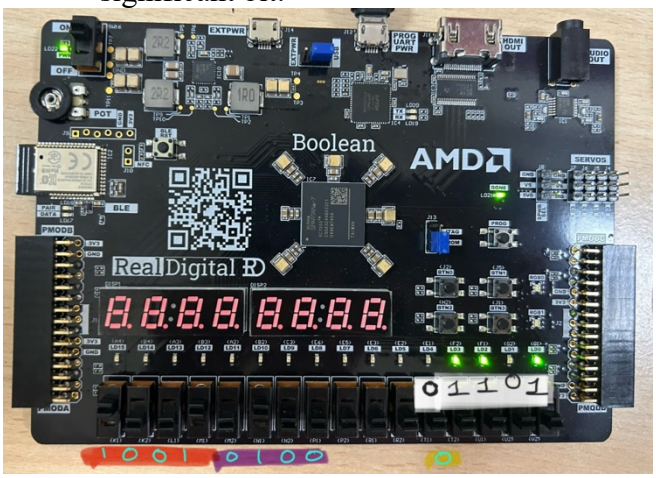


Figure 80 : FPGA example 1

- Example 2 : a = 13(red), b = 12 (purple), c = 0 (yellow). Output, c4 = 1 , s = 1001 (green LEDs) meaning 25. Leftmost bit is most significant bit.

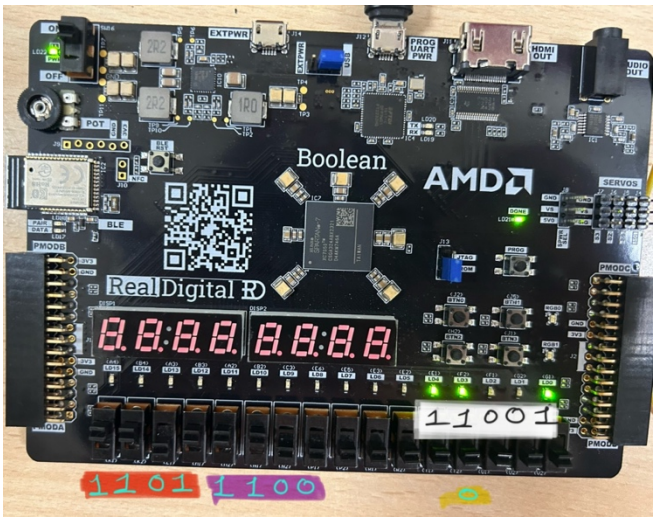


Figure 80 : FPGA example 2

- Google Drive Link for video : https://drive.google.com/drive/folders/1-GoEt94wHtNIwqL6Jj2ryaqdvv-1UmAS

XIII. REFERENCES

1. CMOS VLSI Design : Weste & Harris
2. Digital Integrated Circuits : Jan Rabaey
3. Geeksforgeeks.com