CARRY LOOK AHEAD ADDER

ANUMULA VENKATA SAI SREE SAHITHI
2023112002
IIIT HYDERABAD
sahithi.anumula@research.iiit.ac.in

***Abstract*— The project involves the design and implementation of a 4-bit Carry Look Ahead (CLA) adder, a high-speed arithmetic circuit used for efficient binary addition. The CLA adder is composed of modular blocks for propagating and generating signal computation, carry look-ahead logic, and the sum generation block. *T*he designs ensures that input bits are provided before the rising edge of the clock, with outputs computed and available at the subsequent rising edge, adhering to a synchronous timing scheme**

***Keywords*— Adder, PTL, TSPC, CLA, Propagator, generator, Flipflop**

I. CARRY LOOK AHEAD ADDER

A Carry Look-Ahead (CLA) adder is an advanced type of digital adder designed to improve the speed of binary addition by minimizing the propagation delay associated with ripple-carry adders. In a conventional ripple-carry adder, the carry-out of each bit must wait for the carry-in to be computed from the preceding bit, leading to a delay that grows linearly with the number of bits.

The CLA adder addresses this issue by computing the carry signals in parallel using the concepts of **propagate** (p_i) and **generate** (g_i) functions for each bit. These functions are defined as:

$$p_i = a_i \oplus b_i$$

$$g_i = a_i . b_i$$

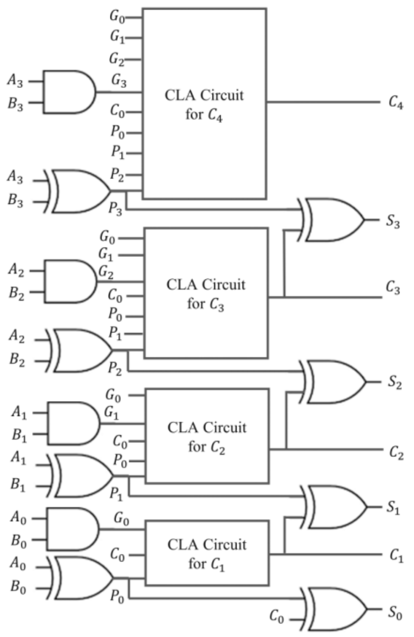
The carry out ($c_{(i+1)}$) of the ith bit position can be written as (assuming $c_0 = 0$) follows:

$$c_{(i+1)} = (p_i . c_i) + g_i \text{ where } (i = 1, 2, 3, 4)$$

This parallelism significantly enhances the adder's speed, making it a preferred choice in high-performance applications like processors and digital signal processing systems.

1.PROPOSED STRUCTURE OF ADDER

The proposed structure for my Carry Look-Ahead (CLA) adder is a 4-bit design using static CMOS logic. This design relies on carry-generate (G_i)and carry-propagate (P_i) terms to calculate the carry-out. These terms are derived from the input bits (Ai and Bi), enabling faster carry calculation compared to ripple carry adders, which reduces delay. The carry-out signals C1, C2, C3, and C4 are generated from four separate blocks, ensuring efficient carry computation. The design eliminates intermediate logic gates, reducing dynamic power consumption and propagation delay. The carry-out circuits are constructed using simplified Boolean equations, ensuring a symmetric and efficient transistor-level implementation. The sum is calculated using XOR gates, making the design both effective and reliable. This structure improves speed and power efficiency while maintaining accurate functionality



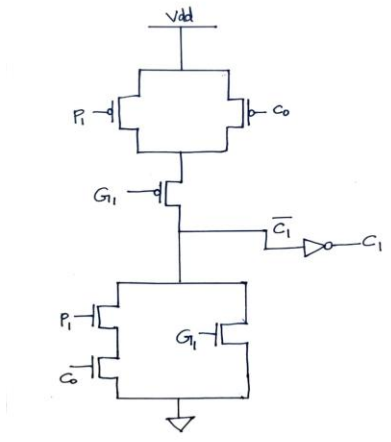
BASIC MODEL OF THE CLA

$$C_{i+1} = G_i + P_iC_i$$

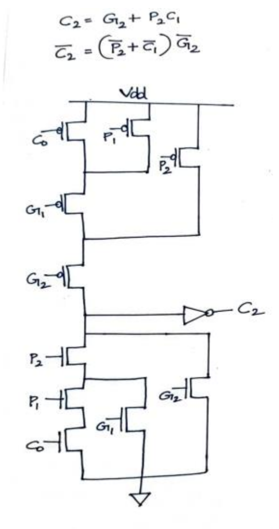
$$C_1 = G_0 + P_0C_0$$

$$C_1 = G_1 + P_1 C_0$$

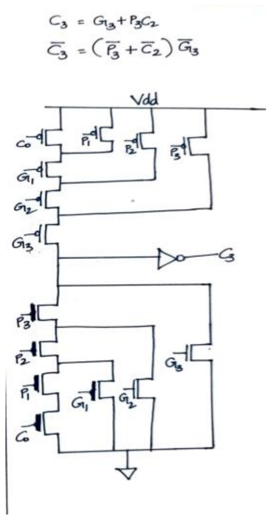
$$\overline{C_1} = (\overline{P_1} + \overline{C_0})\,\overline{G_1}$$



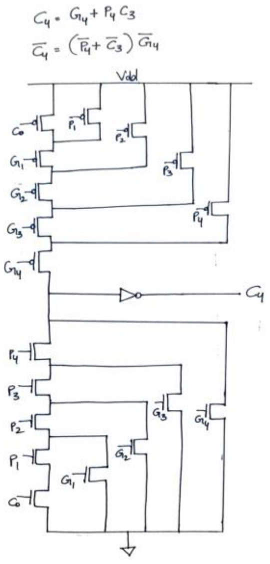
$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$



$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$



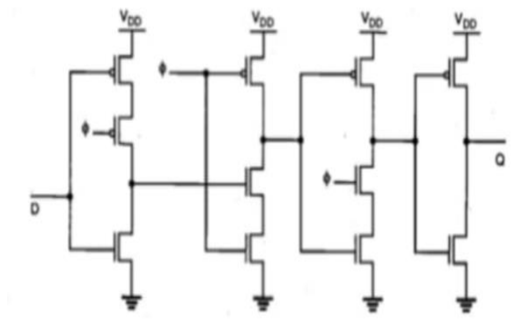
$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$



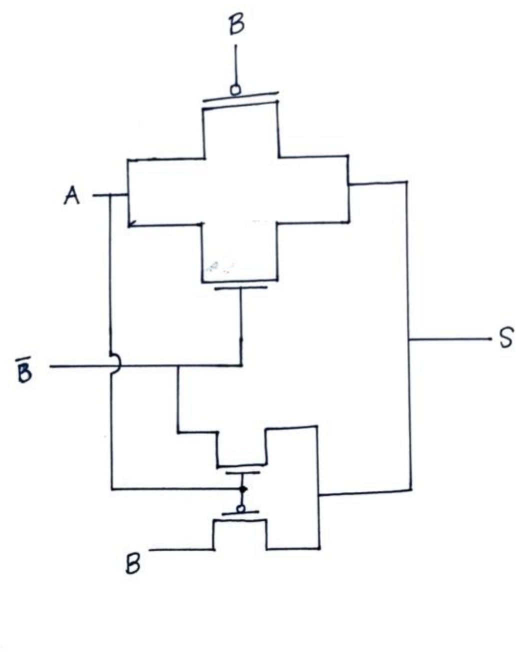
2. DESIGN DETAILS

The design of the 4-bit Carry Look-Ahead (CLA) adder consists of several blocks with specific topologies and sizing:

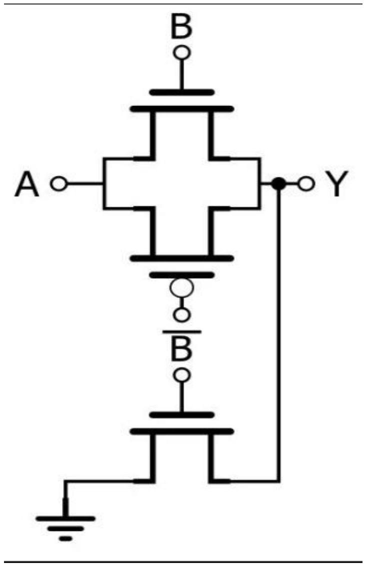
- **D-Flip-Flop**: The flip-flop is implemented using **True Single-Phase Clocked (TSPC)** logic. This choice ensures high-speed operation with reduced clock skew, enabling reliable edge-triggered storage of the carry-in signal.



- **XOR and AND Gates**: The XOR and AND gates, used to generate the carry-propagate (P_i) and carry-generate (G_i) terms, are designed using **Pass-Transistor Logic (PTL)**. PTL is chosen for its compact design and efficiency, helping to minimize area and improve speed.



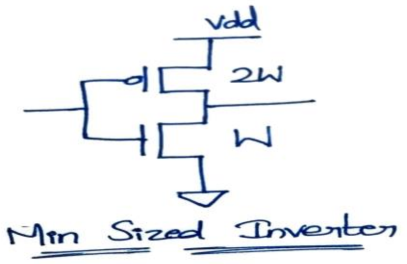
XOR GATE

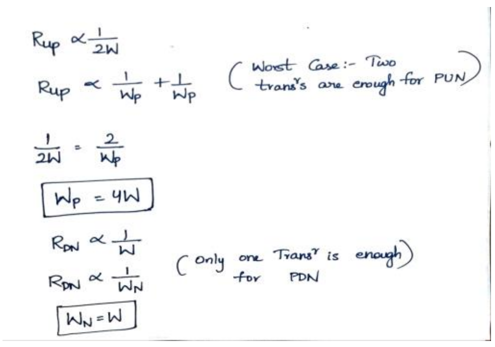


AND GATE

- **Carry Blocks (C1, C2, C3, C4)**: The carry-out signals C1, C2, C3, C4 are generated using **CMOS logic**. Each carry block computes its respective carry-out using the carry-in from the previous block, along with the carry-generate (G_i) and carry-propagate (P_i) signals, ensuring a fast and efficient design.

  The sizing of Carry blocks is done in accordance with a minimum sized inverter





$R_{up} \propto \frac{1}{2W}$

$R_{up} \propto \frac{1}{W_p} + \frac{1}{W_p}$ (Worst Case :- Two trans's are enough for PUN)

$\frac{1}{2W} = \frac{2}{W_p}$

$\boxed{W_p = 4W}$

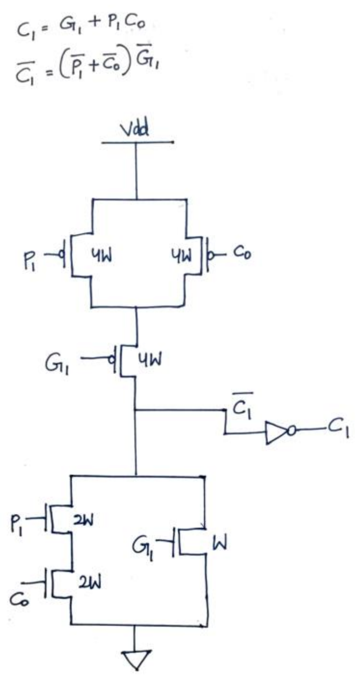
$R_{DN} \propto \frac{1}{W}$

$R_{DN} \propto \frac{1}{W_N}$ (Only one Trans^r is enough for PDN)

$\boxed{W_N = W}$

The sizing of the Carry blocks is done in accordance with a minimum-sized inverter to ensure uniformity and optimal performance. The transistors in the Carry blocks are sized to balance speed and power consumption while maintaining sufficient drive strength to propagate the carry signal efficiently. This sizing ensures that the delay through each Carry block is minimized, contributing to the overall speed of the design.

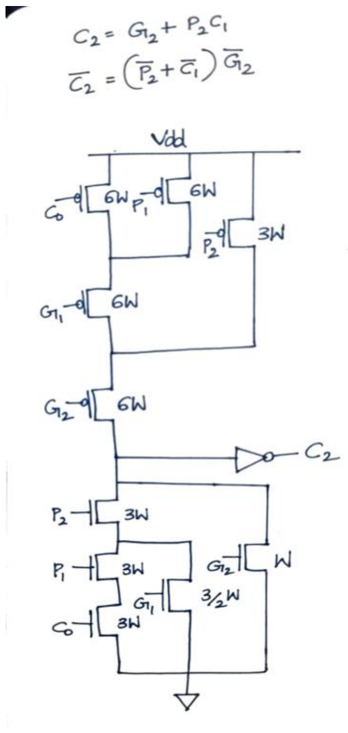
C1 BLOCK



$C_1 = G_1 + P_1 C_0$

$\overline{C_1} = (\overline{P_1} + \overline{C_0})\overline{G_1}$

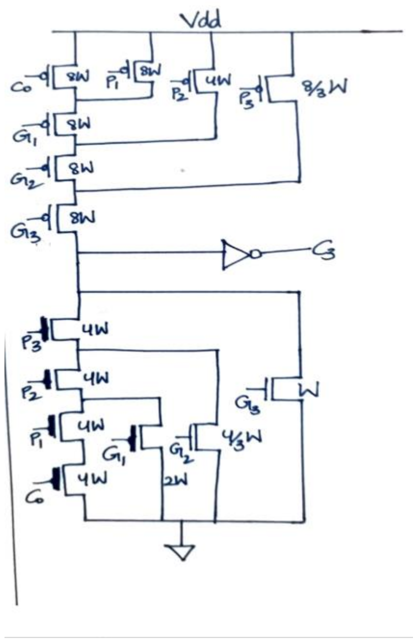
C2 BLOCK



$C_2 = G_2 + P_2 C_1$

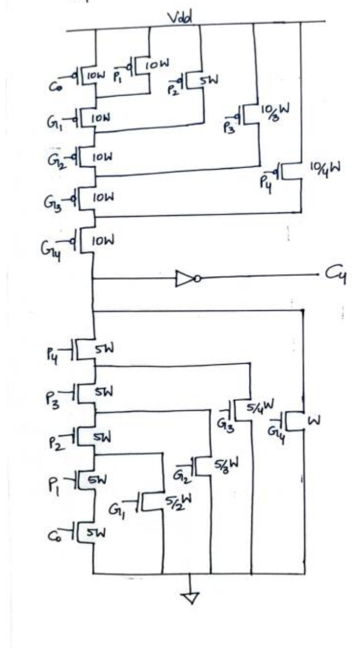
$\overline{C_2} = (\overline{P_2} + \overline{C_1})\overline{G_2}$

$$C_3 = G_3 + P_3 C_2$$
$$\overline{C_3} = (\overline{P_3} + \overline{C_2})\,\overline{G_3}$$



C3 BLOCK

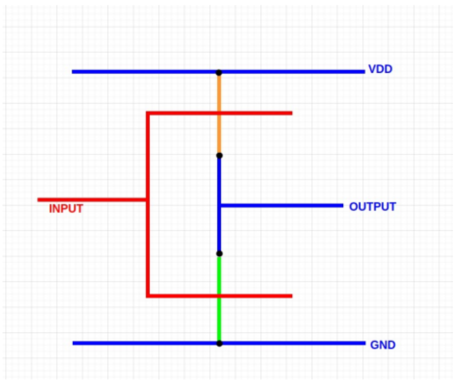
$$C_4 = G_4 + P_4 C_3$$
$$\overline{C_4} = (\overline{P_4} + \overline{C_3})\,\overline{G_4}$$



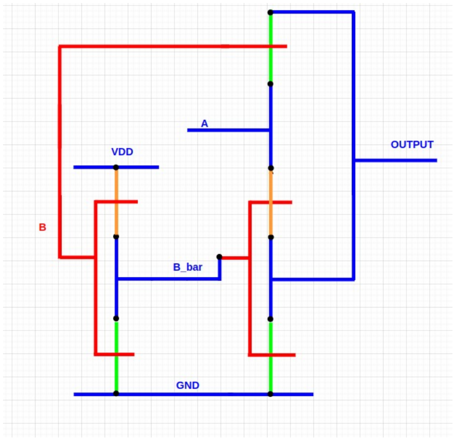
C4 BLOCK

All blocks are sized and optimized using **TSMC 180nm process parameters**. The PMOS and NMOS transistors in the CMOS logic for the carry blocks are carefully sized to minimize delay while maintaining sufficient drive strength. The PTL gates are also optimized for low resistance and fast switching speeds, especially in the XOR and AND gates. The overall design combines TSPC flip-flops for high-speed storage, PTL for efficient logic gate design, and CMOS logic for carry generation.

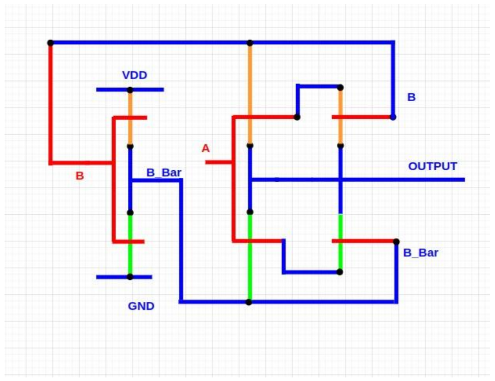
3. STICK DIAGRAMS



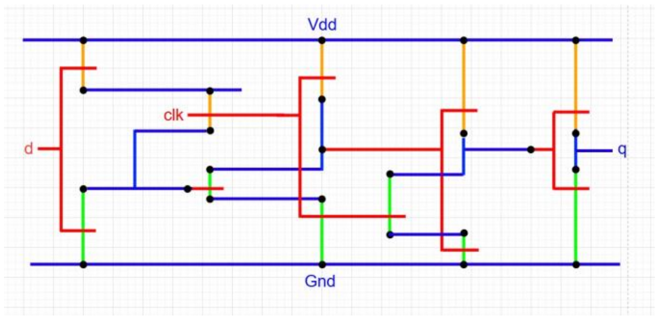
INVERTER



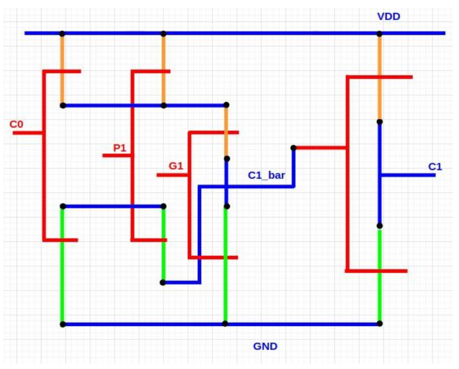
AND GATE



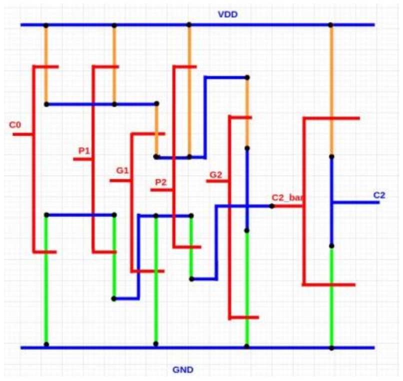
XOR GATE



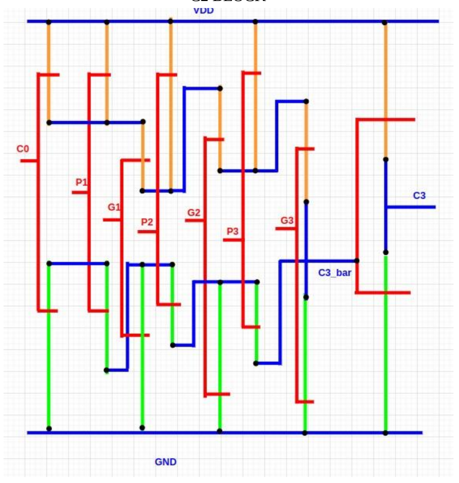
FLIPFLOP



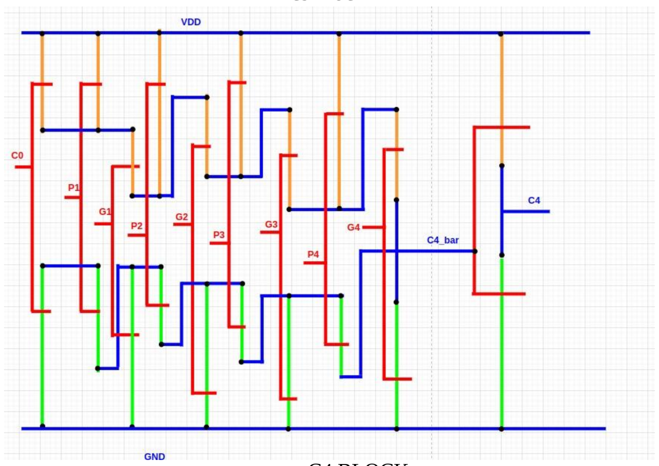
C1 BLOCK



C2 BLOCK



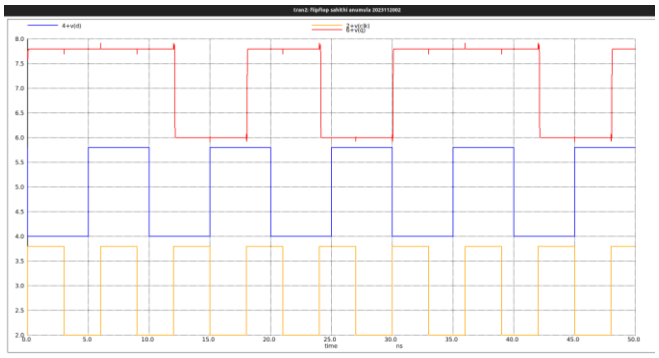
C3 BLOCK



C4 BLOCK

4. NGSPICE PRE-LAYOUT

TSPC Positive Edge Flipflop:

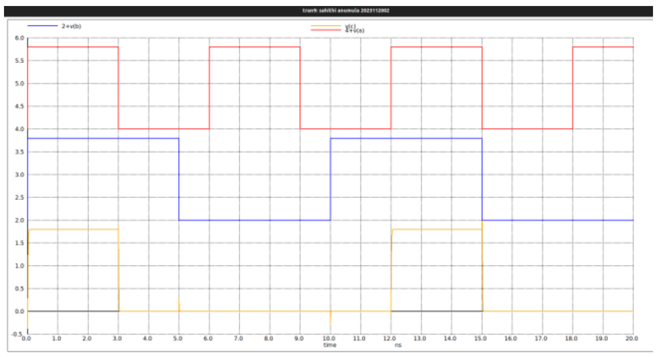


Inputs:

Vclk clk gnd pulse 0 1.8 0n 0 0 3n 6n

Vd d gnd pulse 1.8 0 0n 0 0 5n 10n

AND GATE (PTL LOGIC)

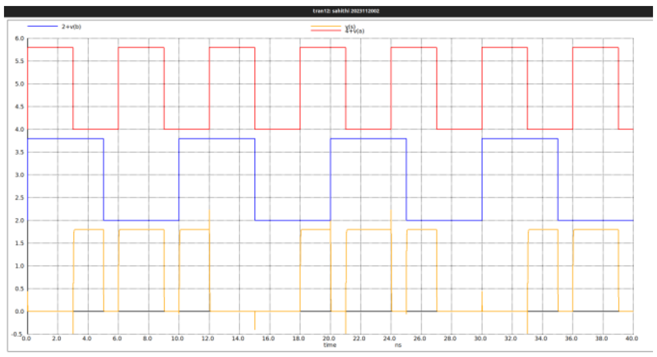


Inputs:

vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 3ns 6ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 5ns 10ns

XOR GATE (PTL LOGIC):

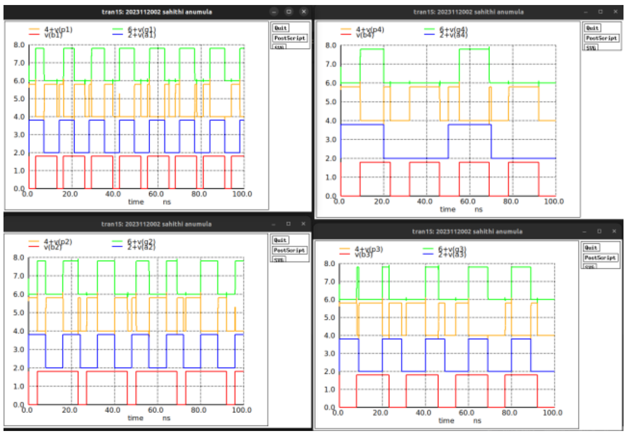


Inputs:

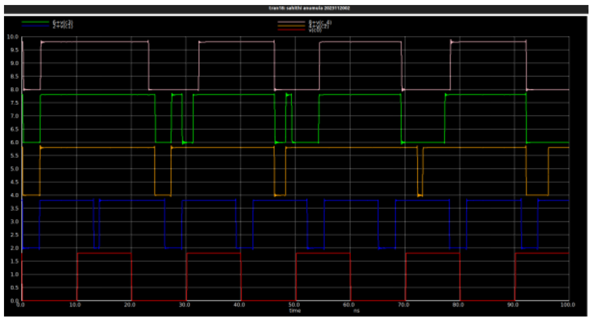
vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 3ns 6ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 5ns 10ns

Propagate and generate block



CLA BLOCK



Inputs:

vin_a1 p1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns

vin_a2 p2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns

vin_a3 p3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

vin_a4 p4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 50ns

vin_b1 g1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns

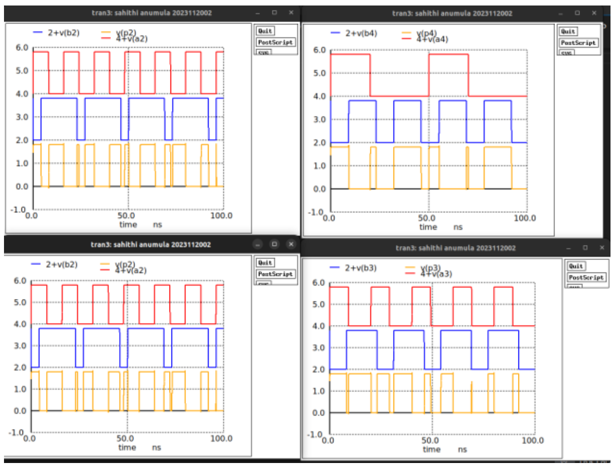
vin_b2 g2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_b3 g3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

vin_b4 g4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns

vin_c0 c0 0 pulse (1.8 0 0 0 0 10n 20n)

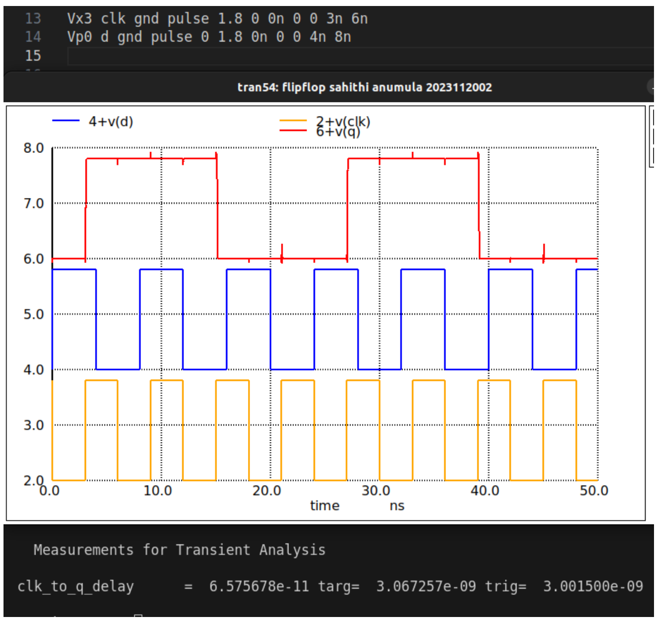
Sum block



5. FLIP FLOP HOLD TIME AND SET UP TIME

It has the set up time of 0.081 ns

It has hold time of 0.012 ns

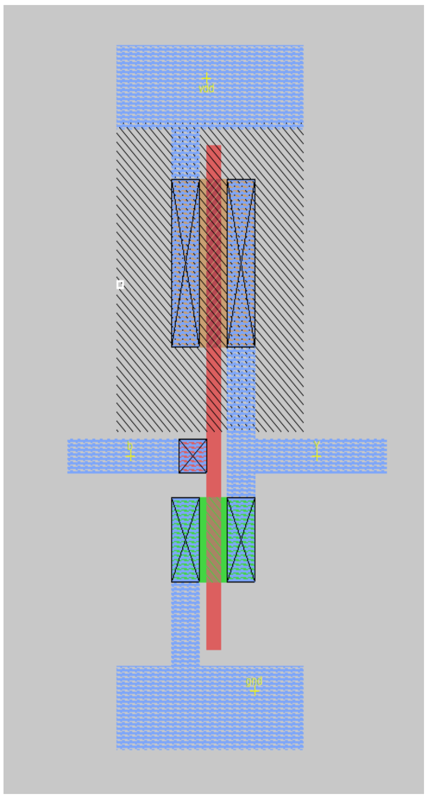


```
Vx3 clk gnd pulse 1.8 0 0n 0 0 3n 6n
Vp0 d gnd pulse 0 1.8 0n 0 0 4n 8n
```

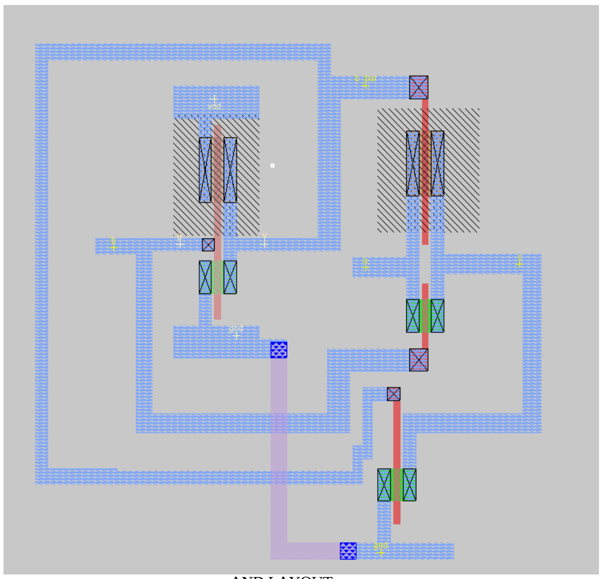
```
  Measurements for Transient Analysis

clk_to_q_delay      =  6.575678e-11 targ=  3.067257e-09 trig=  3.001500e-09
```

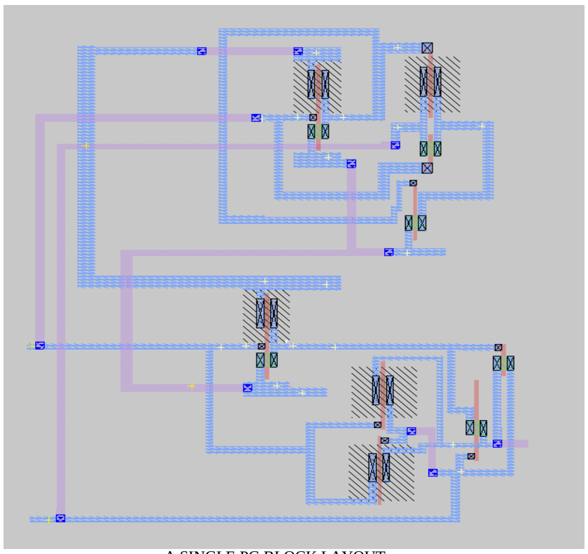
Clk to q delay = 6.575678e-11

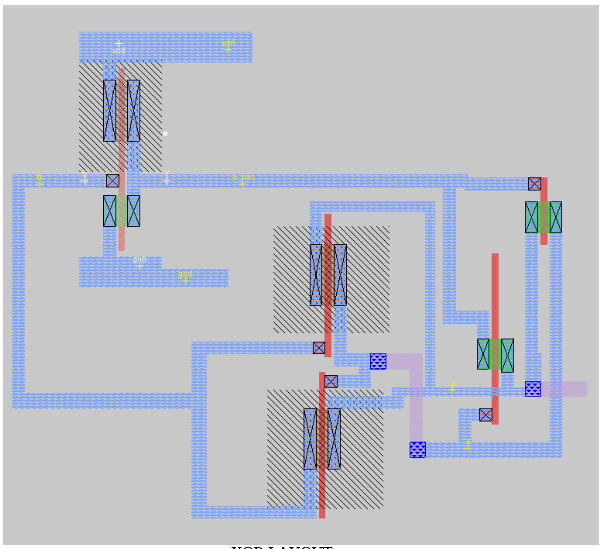
6. MAGIC LAYOUTS AND POST LAYOUTS

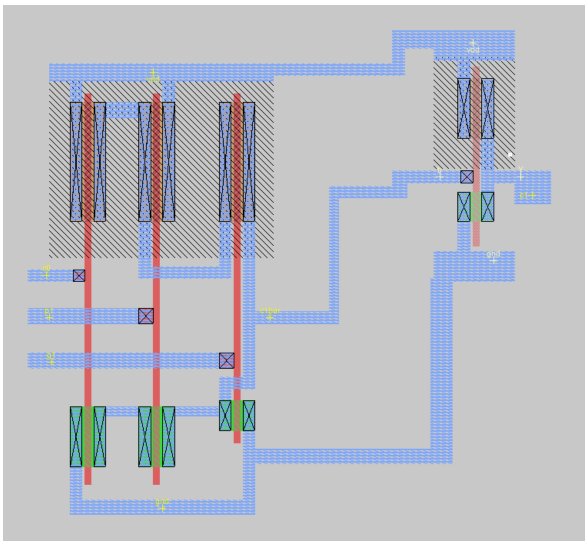


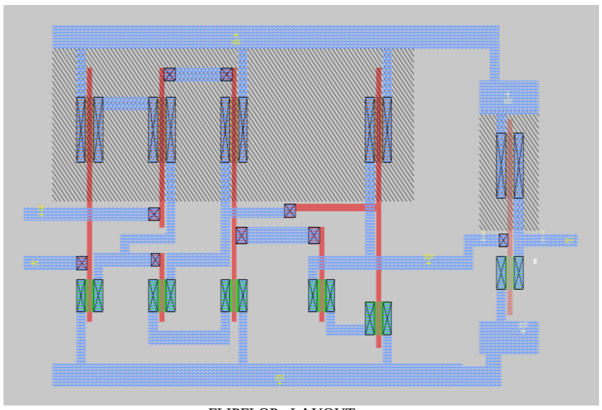
INVERTER LAYOUT

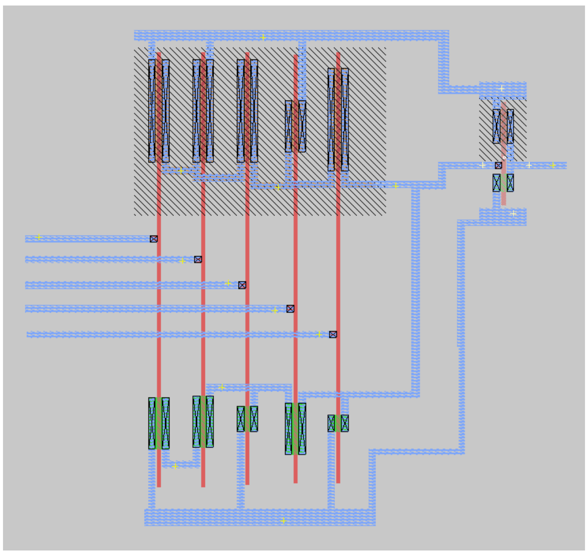

AND LAYOUT

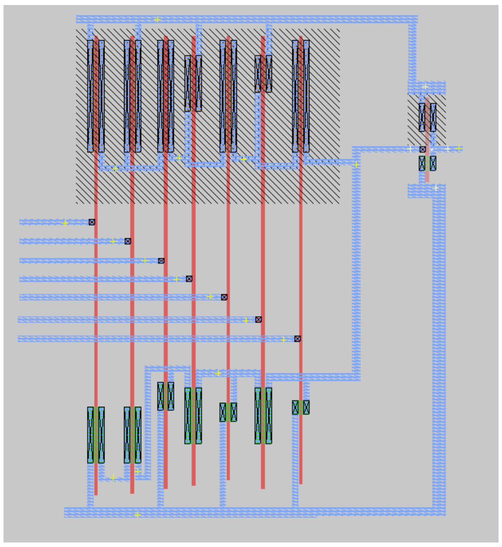

A SINGLE PG BLOCK LAYOUT

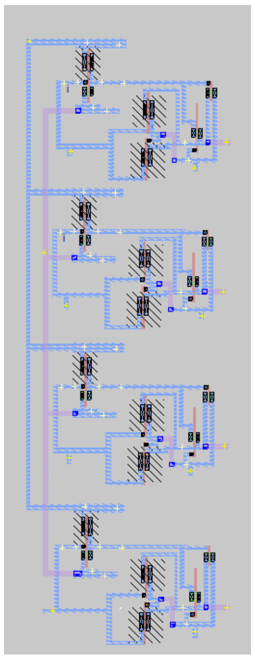

XOR LAYOUT

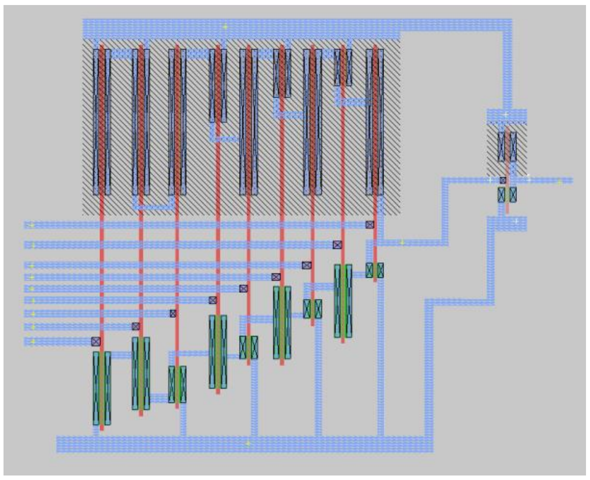

C1 BLOCK LAYOUT

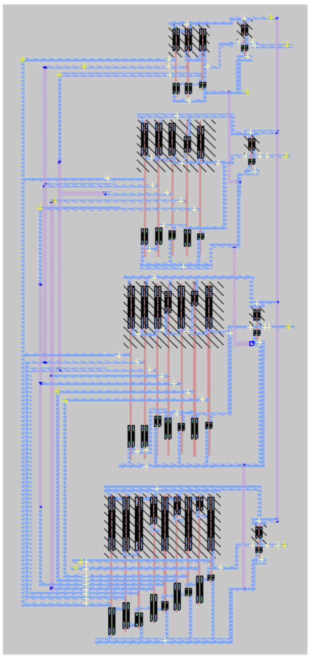

FLIPFLOP   LAYOUT

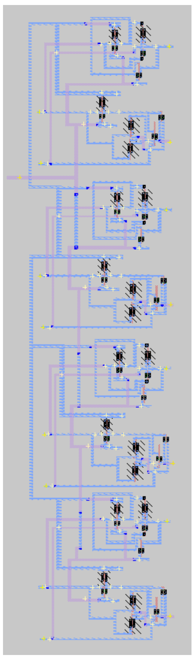

C2 BLOCK   LAYOUT

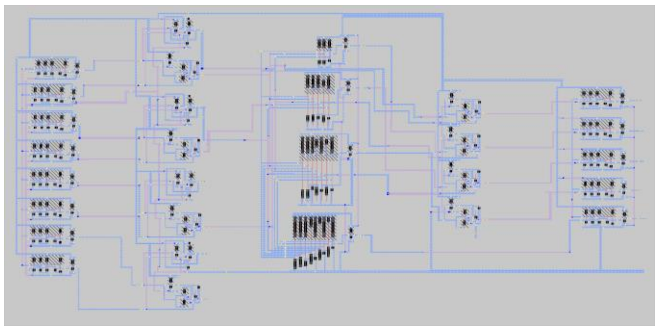

C3 BLOCK   LAYOUT


SUM BLOCK   LAYOUT

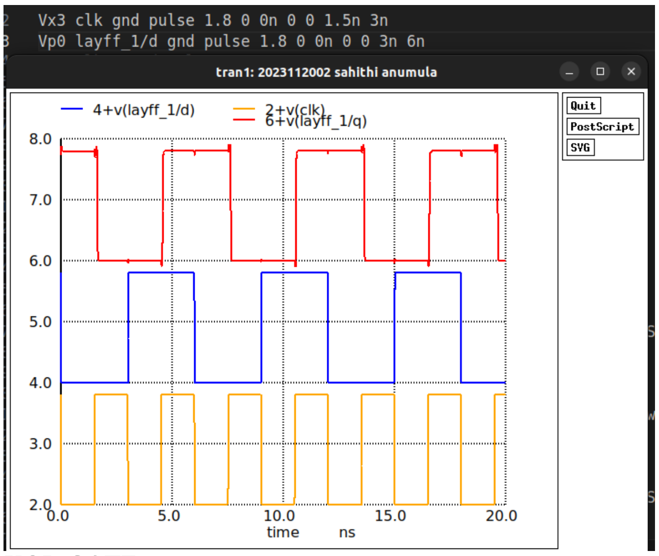

C4 BLOCK   LAYOUT


CLA BLOCK   LAYOUT


PG BLOCK   LAYOUT

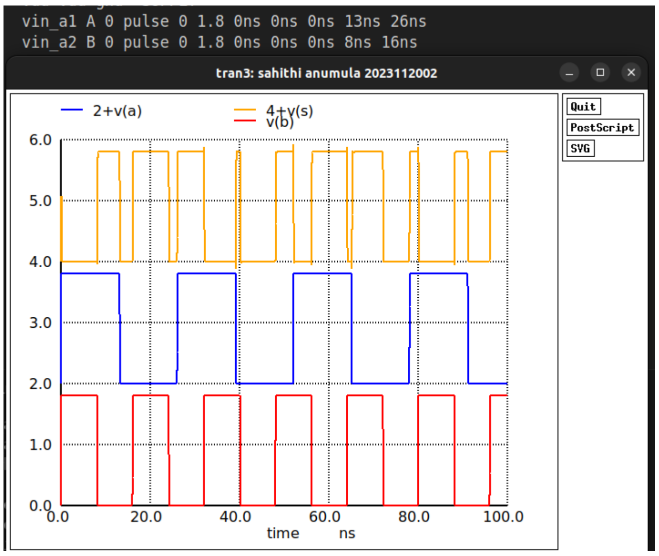

FINAL   LAYOUT

TSPC FLIPFLOP: -



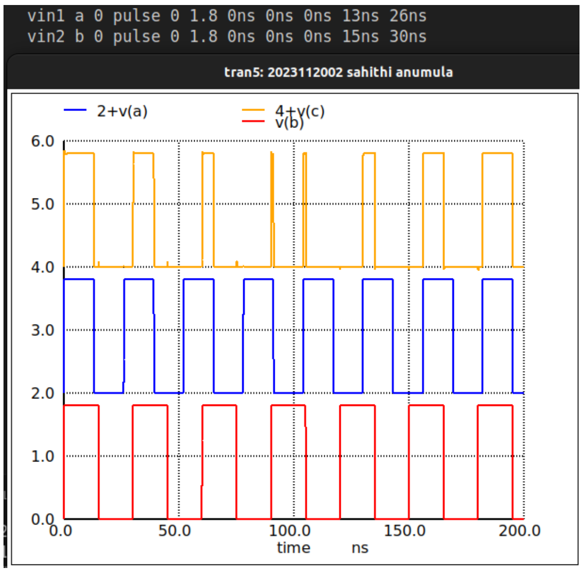
```
Vx3 clk gnd pulse 1.8 0 0n 0 0 1.5n 3n
Vp0 layff_1/d gnd pulse 1.8 0 0n 0 0 3n 6n
```

XOR GATE: -



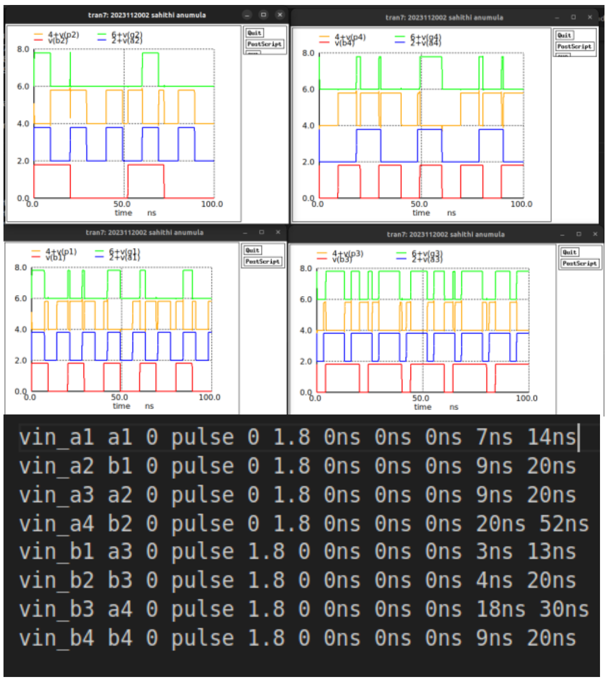
```
vin_a1 A 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns
vin_a2 B 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns
```

AND GATE: -



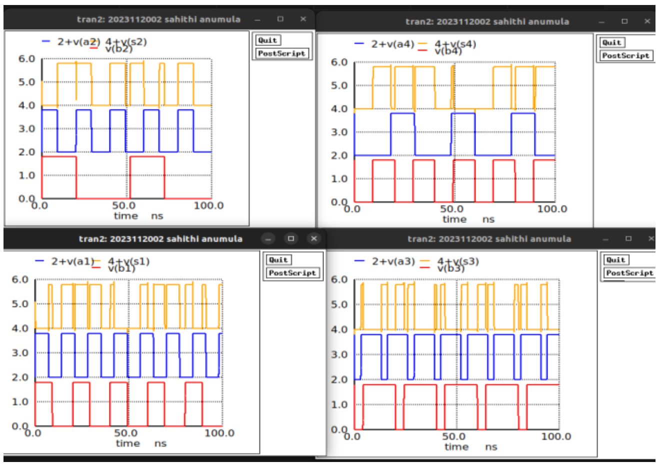
```
vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns
vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns
```

PG BLOCK

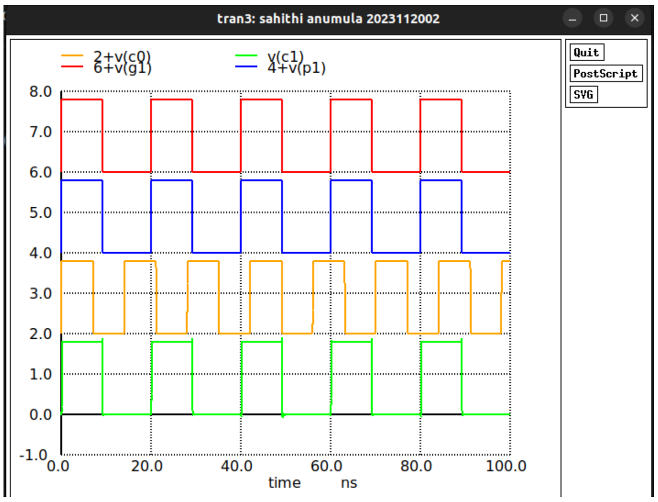


```
vin_a1 a1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns
vin_a2 b1 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a3 a2 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a4 b2 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns
vin_b1 a3 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns
vin_b2 b3 0 pulse 1.8 0 0ns 0ns 0ns 4ns 20ns
vin_b3 a4 0 pulse 1.8 0 0ns 0ns 0ns 18ns 30ns
vin_b4 b4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 20ns
```

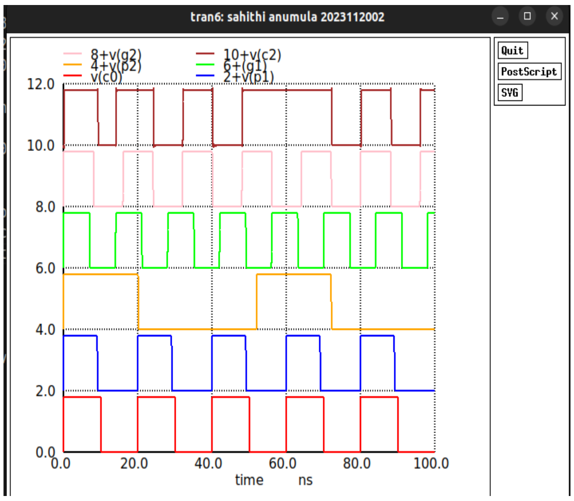
SUM BLOCK:



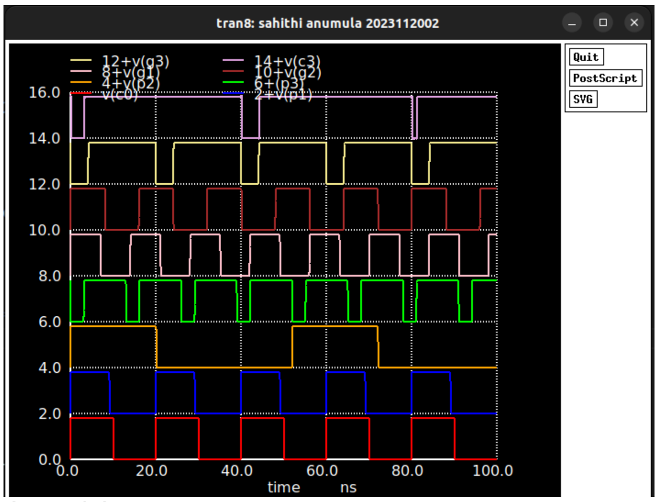
C1 BLOCK:



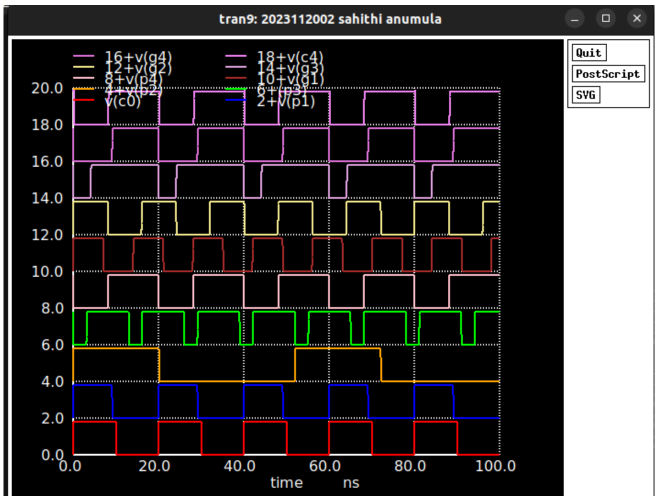
C2 BLOCK:



C3 BLOCK:



C4 BLOCK:



The above blocks follow the logic mentioned below

$C_1 = G_0 + P_0C_0$

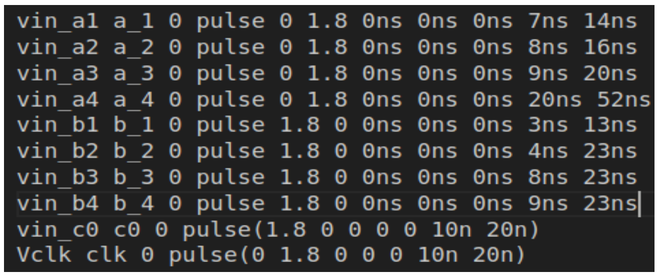
$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

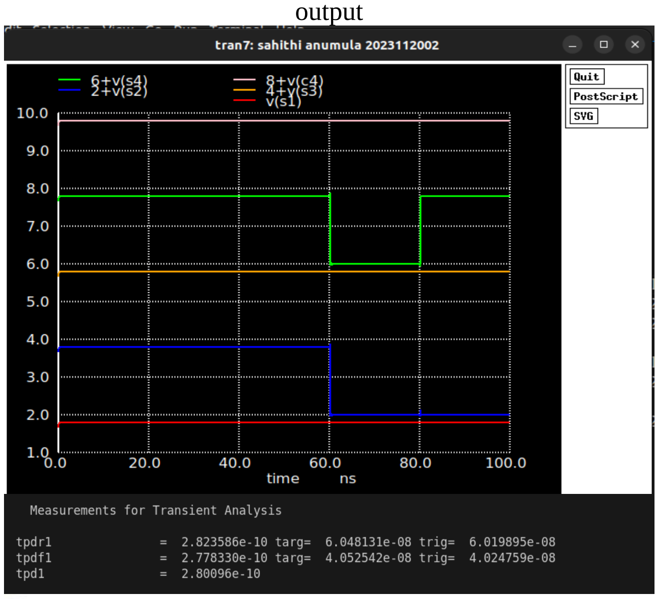
The final output and values are under part 9

7.NGSPICE FINAL PRELAYOUT



```
vin_a1 a_1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns
vin_a2 a_2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns
vin_a3 a_3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a4 a_4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns
vin_b1 b_1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns
vin_b2 b_2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns
vin_b3 b_3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns
vin_b4 b_4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns
vin_c0 c0 0 pulse(1.8 0 0 0 0 10n 20n)
Vclk clk 0 pulse(0 1.8 0 0 0 10n 20n)
```

output



```
Measurements for Transient Analysis

tpdr1               =  2.823586e-10 targ=  6.048131e-08 trig=  6.019895e-08
tpdf1               =  2.778330e-10 targ=  4.052542e-08 trig=  4.024759e-08
tpd1                =  2.80096e-10
```

$$T_{clk} \geq t_{pcq} + t_{pd} + t_{su}$$

$$T_{clk\ min} = Freq\ clk\ max$$

$$t_{pcq} = 6.57 \times 10^{-11}$$

$$t_{pd} = 2.9008 \times 10^{-10}$$

$$t_{su} = 8.1 \times 10^{-11}$$

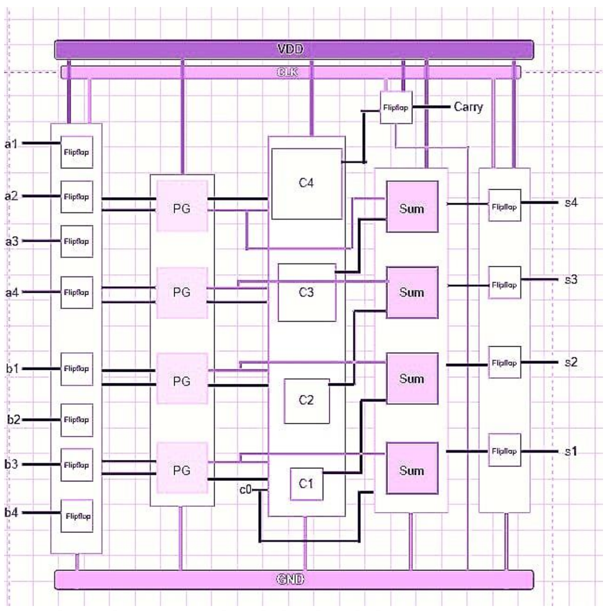
$$T_{clk\ min} = 42.678 \times 10^{-11}$$

$$Max\ clk = 2.3 \times 10^{9}\ Hz$$

$$= 2.3GHz$$

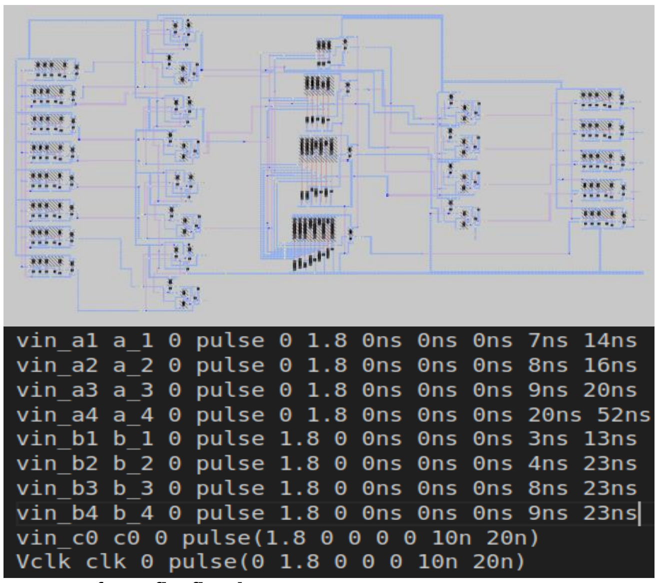
The clock frequency is 2.3GHz

8.FLOOR PLAN



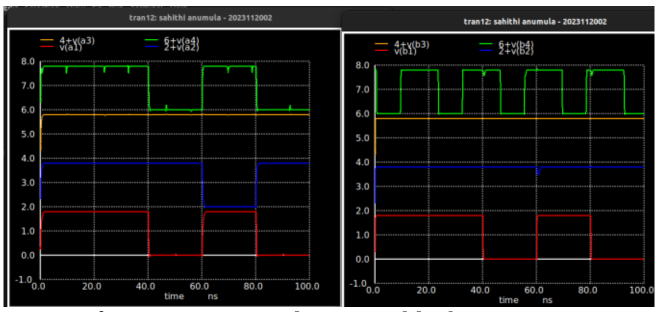
Vertical Pitch = 1477*lambda (lambda = 0.09) = 132.93n
Horizontal Pitch = 3135*lambda (lambda = 0.09) = 282.15n

9. FINAL POST LAYOUT RESULTS

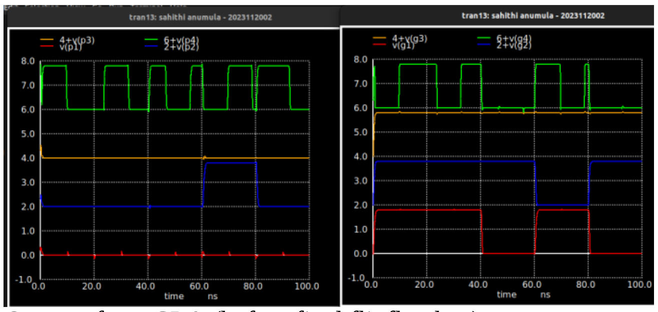


```
vin_a1 a_1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns
vin_a2 a_2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns
vin_a3 a_3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a4 a_4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns
vin_b1 b_1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns
vin_b2 b_2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns
vin_b3 b_3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns
vin_b4 b_4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns
vin_c0 c0 0 pulse(1.8 0 0 0 0 10n 20n)
Vclk clk 0 pulse(0 1.8 0 0 0 10n 20n)
```

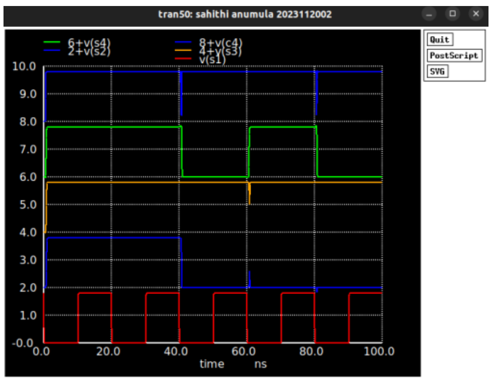
Outputs from flipflop bus



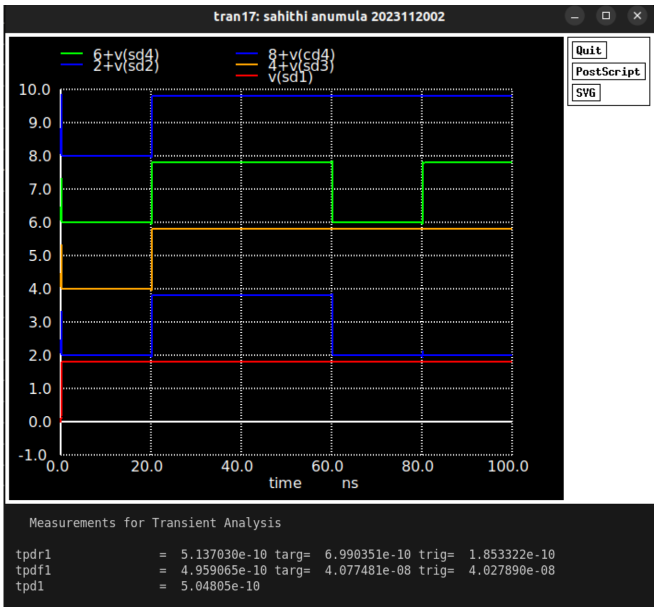
Outputs from propagate and generate block



Outputs from CLA (before final flipflop bus)

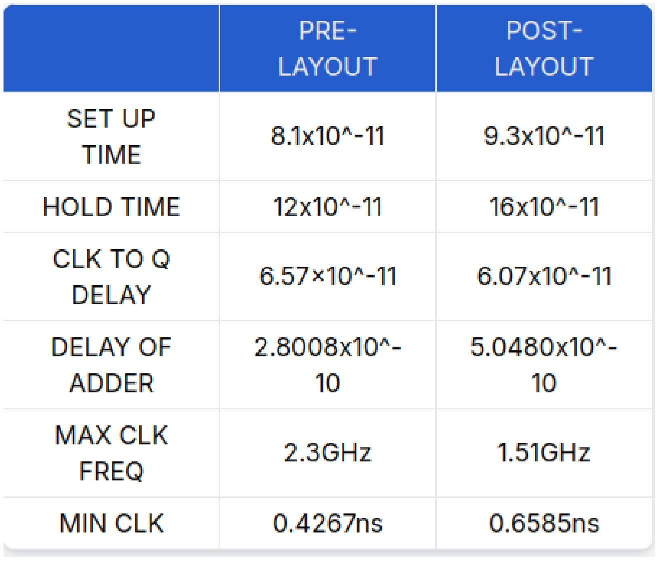


FINAL OUTPUT



```
Measurements for Transient Analysis

tpdr1           = 5.137030e-10 targ= 6.990351e-10 trig= 1.853322e-10
tpdf1           = 4.959065e-10 targ= 4.077481e-08 trig= 4.027890e-08
tpd1            = 5.04805e-10
```

COMPARISION:



	PRE-LAYOUT	POST-LAYOUT
SET UP TIME	8.1x10^-11	9.3x10^-11
HOLD TIME	12x10^-11	16x10^-11
CLK TO Q DELAY	6.57×10^-11	6.07x10^-11
DELAY OF ADDER	2.8008x10^-10	5.0480x10^-10
MAX CLK FREQ	2.3GHz	1.51GHz
MIN CLK	0.4267ns	0.6585ns

10. CLOCK DELAY AND FREQUENCY

The worst-case delay of CLA Adder is 5.04805x10^{-10}

$$T_{clk} \geq t_{pcq} + t_{pd} + t_{su}$$

$$T_{clk\,min} \geq \text{Freq clk max}$$

$$t_{pcq} = 6.07 \times 10^{-11}$$

$$t_{pd} = 5.048 \times 10^{-10}$$

$$t_{su} = 9.3 \times 10^{-11}$$

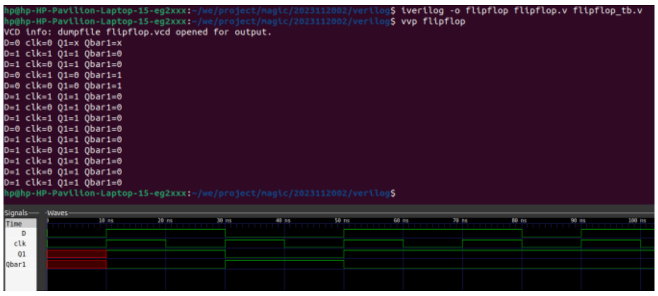
$$T_{clk\,min} \geq 65.85 \times 10^{-11}$$

$$\text{Freq max} \geq 1.51 \text{ GHz}$$

The clock frequency is 1.51GHz

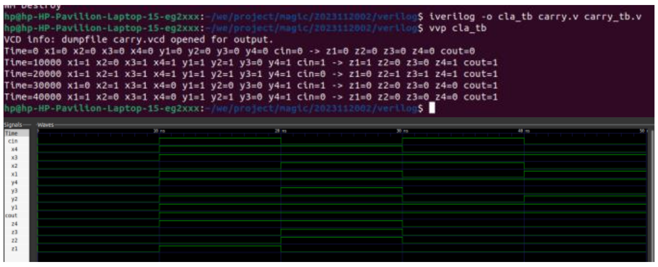
11. VERILOG

FLIPFLOP



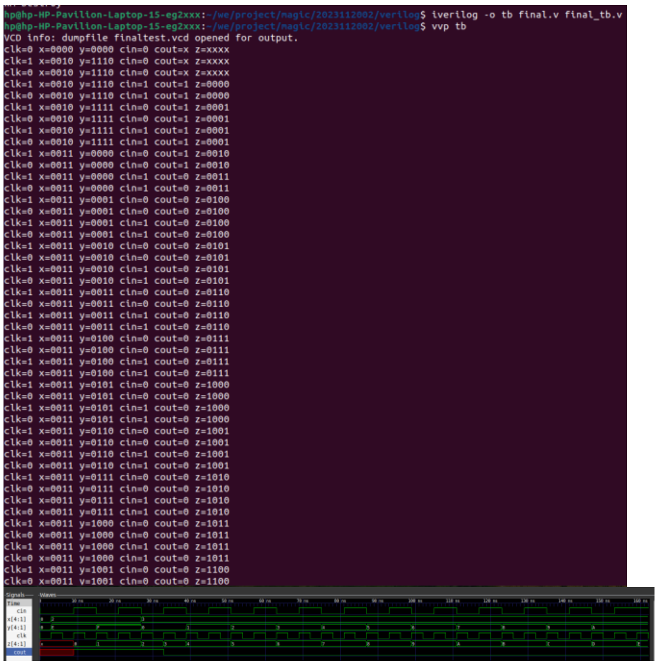
```
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ iverilog -o flipflop flipflop.v flipflop_tb.v
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ vvp flipflop
VCD info: dumpfile flipflop.vcd opened for output.
D=0 clk=0 Q1=x Qbar1=x
D=1 clk=1 Q1=1 Qbar1=0
D=1 clk=0 Q1=1 Qbar1=0
D=0 clk=1 Q1=0 Qbar1=1
D=0 clk=0 Q1=0 Qbar1=1
D=1 clk=1 Q1=1 Qbar1=0
D=1 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=1 Qbar1=0
D=0 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=1 Qbar1=0
D=1 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=1 Qbar1=0
D=1 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=1 Qbar1=0
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$
```

CLA



```
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ iverilog -o cla_tb carry.v carry_tb.v
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ vvp cla_tb
VCD info: dumpfile carry.vcd opened for output.
Time=0 x1=0 x2=0 x3=0 x4=0 y1=0 y2=0 y3=0 y4=0 cin=0 -> z1=0 z2=0 z3=0 z4=0 cout=0
Time=10000 x1=1 x2=0 x3=1 x4=1 y1=1 y2=1 y3=0 y4=1 cin=1 -> z1=1 z2=0 z3=0 z4=1 cout=1
Time=20000 x1=1 x2=1 x3=1 x4=1 y1=1 y2=1 y3=1 y4=1 cin=0 -> z1=0 z2=1 z3=1 z4=1 cout=1
Time=30000 x1=0 x2=1 x3=1 x4=0 y1=1 y2=0 y3=0 y4=1 cin=1 -> z1=0 z2=0 z3=0 z4=0 cout=1
Time=40000 x1=1 x2=0 x3=1 x4=0 y1=1 y2=1 y3=0 y4=1 cin=0 -> z1=0 z2=0 z3=0 z4=0 cout=1
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$
```

Final Output:

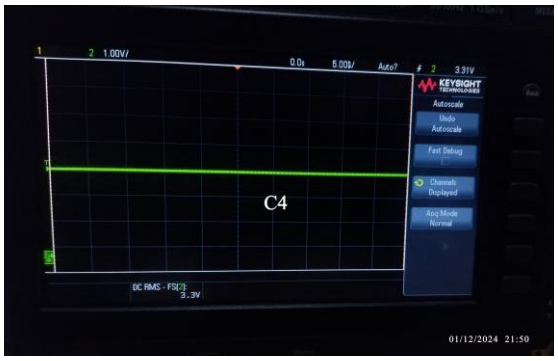


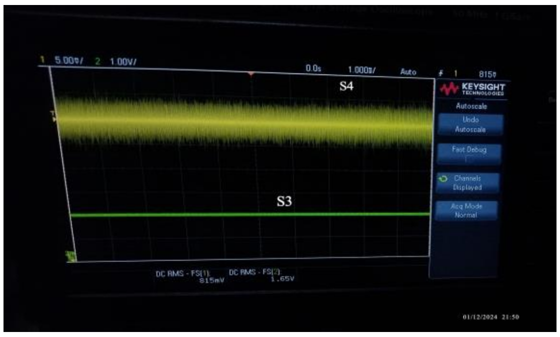
```
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ iverilog -o tb final.v final_tb.v
hp@hp-HP-Pavilion-Laptop-15-eg2xxx:~/we/project/magic/2023112002/verilog$ vvp tb
VCD info: dumpfile finaltest.vcd opened for output.
clk=0 x=0000 y=0000 cin=0 cout=x z=xxxx
clk=1 x=0010 y=1110 cin=0 cout=x z=xxxx
clk=0 x=0010 y=1110 cin=0 cout=x z=xxxx
clk=1 x=0010 y=1110 cin=1 cout=1 z=0000
clk=0 x=0010 y=1110 cin=1 cout=1 z=0000
clk=1 x=0010 y=1111 cin=0 cout=1 z=0001
clk=0 x=0010 y=1111 cin=0 cout=1 z=0001
clk=1 x=0010 y=1111 cin=1 cout=1 z=0001
clk=0 x=0010 y=1111 cin=1 cout=1 z=0001
clk=1 x=0011 y=0000 cin=0 cout=1 z=0010
clk=0 x=0011 y=0000 cin=0 cout=1 z=0010
clk=1 x=0011 y=0000 cin=1 cout=0 z=0011
clk=0 x=0011 y=0000 cin=1 cout=0 z=0011
clk=1 x=0011 y=0001 cin=0 cout=0 z=0100
clk=0 x=0011 y=0001 cin=0 cout=0 z=0100
clk=1 x=0011 y=0001 cin=1 cout=0 z=0100
clk=0 x=0011 y=0001 cin=1 cout=0 z=0100
clk=1 x=0011 y=0010 cin=0 cout=0 z=0101
clk=0 x=0011 y=0010 cin=0 cout=0 z=0101
clk=1 x=0011 y=0010 cin=1 cout=0 z=0101
clk=0 x=0011 y=0010 cin=1 cout=0 z=0101
clk=1 x=0011 y=0011 cin=0 cout=0 z=0110
clk=0 x=0011 y=0011 cin=0 cout=0 z=0110
clk=1 x=0011 y=0011 cin=1 cout=0 z=0110
clk=0 x=0011 y=0011 cin=1 cout=0 z=0110
clk=1 x=0011 y=0100 cin=0 cout=0 z=0111
clk=0 x=0011 y=0100 cin=0 cout=0 z=0111
clk=1 x=0011 y=0100 cin=1 cout=0 z=0111
clk=0 x=0011 y=0100 cin=1 cout=0 z=0111
clk=1 x=0011 y=0101 cin=0 cout=0 z=1000
clk=0 x=0011 y=0101 cin=0 cout=0 z=1000
clk=1 x=0011 y=0101 cin=1 cout=0 z=1000
clk=0 x=0011 y=0101 cin=1 cout=0 z=1000
clk=1 x=0011 y=0110 cin=0 cout=0 z=1001
clk=0 x=0011 y=0110 cin=0 cout=0 z=1001
clk=1 x=0011 y=0110 cin=1 cout=0 z=1001
clk=0 x=0011 y=0110 cin=1 cout=0 z=1001
clk=1 x=0011 y=0111 cin=0 cout=0 z=1010
clk=0 x=0011 y=0111 cin=0 cout=0 z=1010
clk=1 x=0011 y=0111 cin=1 cout=0 z=1010
clk=0 x=0011 y=0111 cin=1 cout=0 z=1010
clk=1 x=0011 y=1000 cin=0 cout=0 z=1011
clk=0 x=0011 y=1000 cin=0 cout=0 z=1011
clk=1 x=0011 y=1000 cin=1 cout=0 z=1011
clk=0 x=0011 y=1000 cin=1 cout=0 z=1011
clk=1 x=0011 y=1001 cin=0 cout=0 z=1100
clk=0 x=0011 y=1001 cin=0 cout=0 z=1100
```

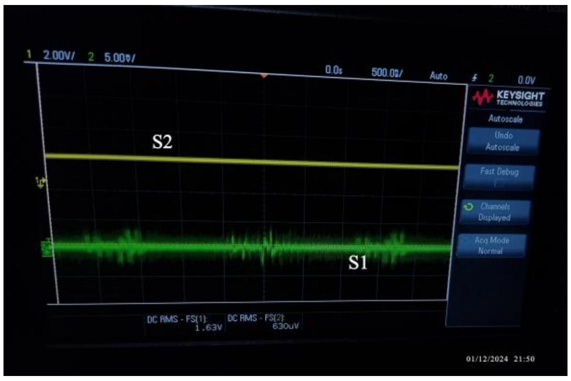
12. FPGA

The chosen bits for this FPGA application are

A = 1 0 0 1   B = 1 1 0 1



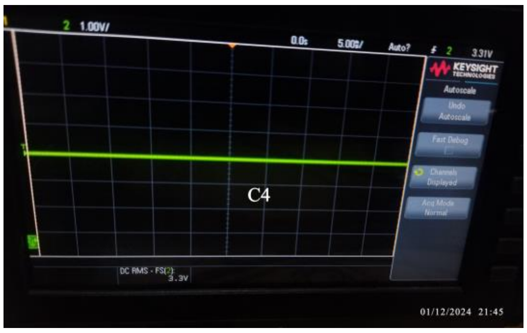


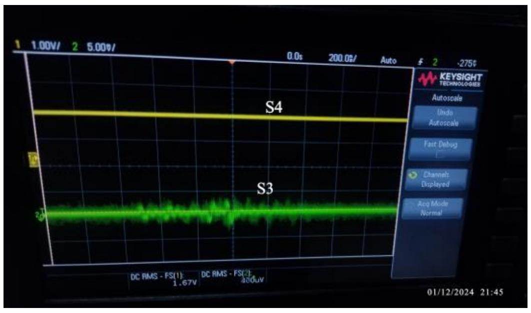


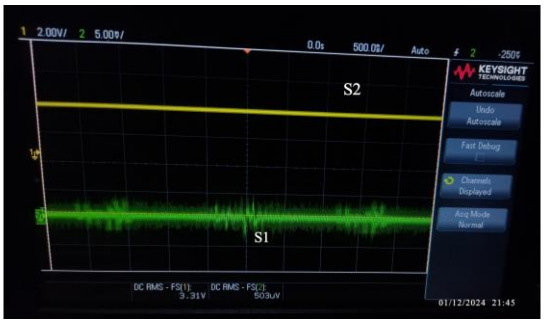
Final solution is 1 0 1 1 0

Case 2:

A = 1 1 0 0   B = 1 1 1 0







Final solution is 1 1 0 1 0