CARRY LOOK AHEAD ADDER

KEERTHI SEELA
2023102012
IIIT HYDERABAD
keerthi.seela@students.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*—Flipflop Buses, Propogate/Generate Block, CLA Block, Sum Block**

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:

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

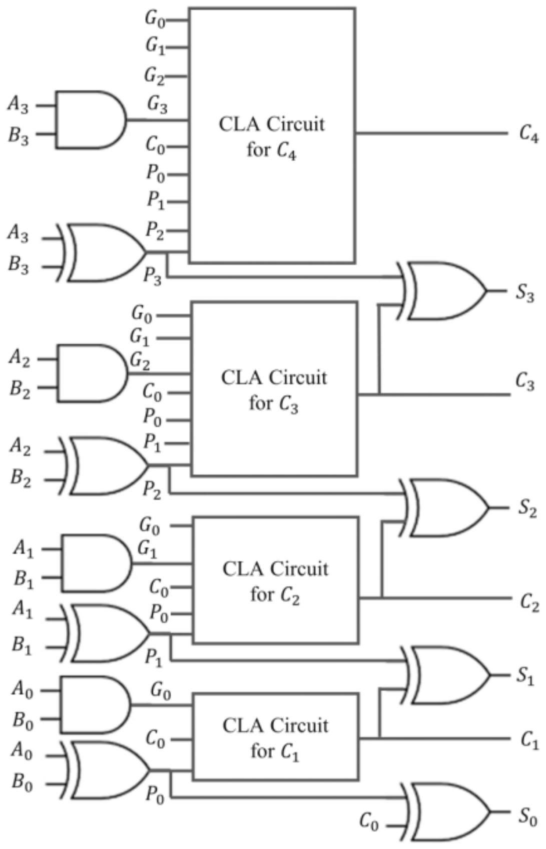


Figure: BASIC MODEL OF THE CLA

$C_1 = G_0 + P_0C_0$

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

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

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

$P_3P_2P_1P_0C_0$

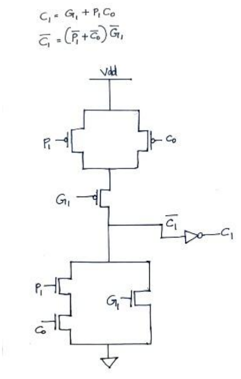


Figure: C-1 Block

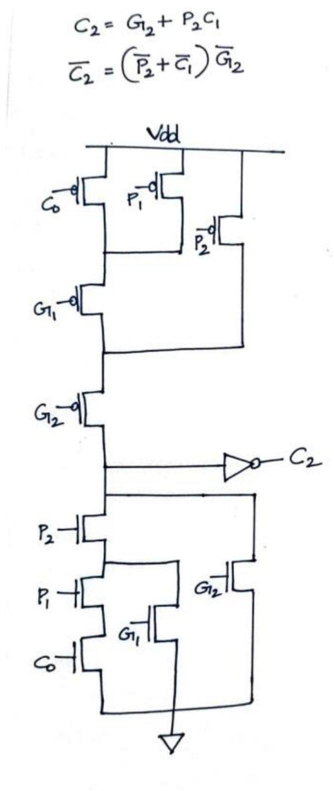


Figure: C-2 Block

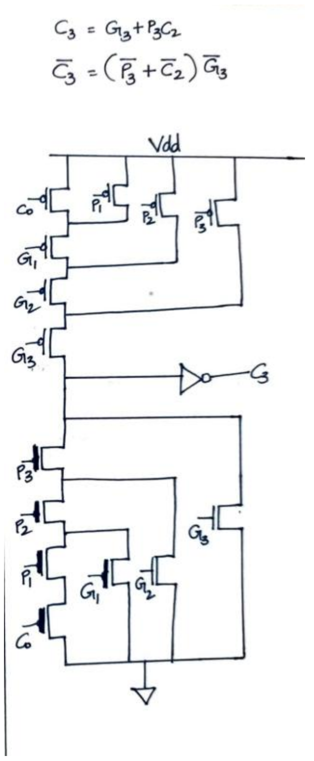


Figure: C-3 Block

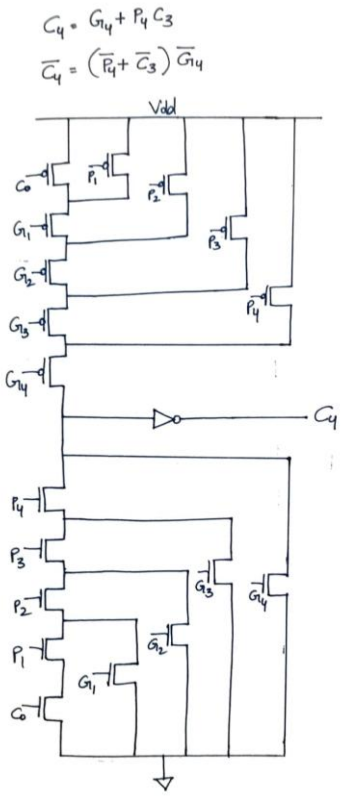


Figure: C-4 Block

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.

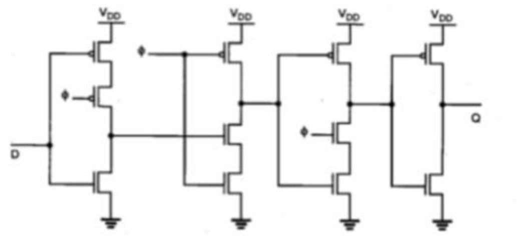


Figure: TSPC D-Flipflop

- **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. These gates are used to build Propogate and Generate block and Sum Block.

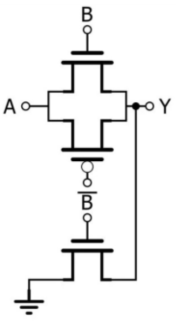


Figure: PTL AND Gate

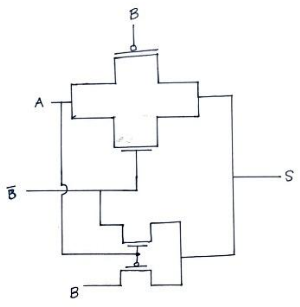
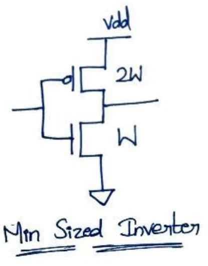
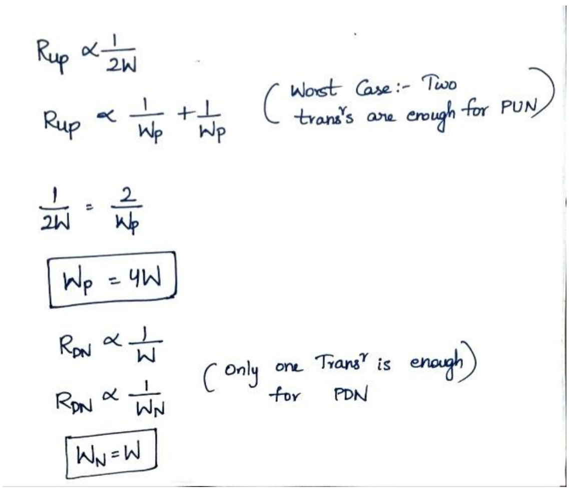


Figure: PTL XOR Gate

- **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.



For C-1 Block:



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

$R_{up} \propto \frac{1}{W_p} + \frac{1}{W_p}$ (Worst Case:- Two trans^r'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)

$\boxed{W_N = W}$

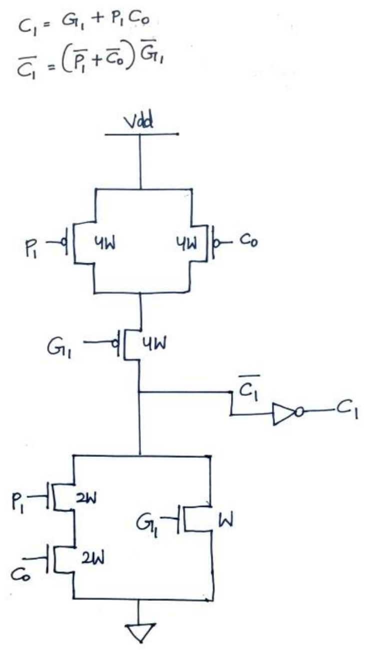


Figure: Sizing of C-1 Block

Similar method is carried down for sizing C2, C3, C4 as well.

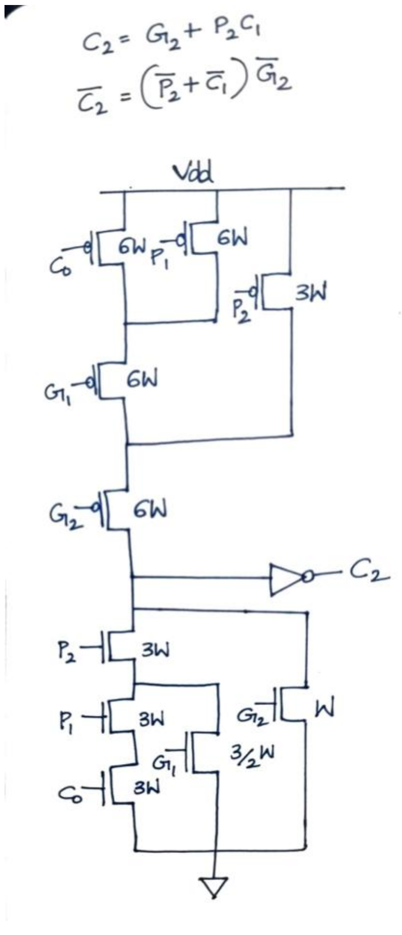


Figure: Sizing of C-2 Block

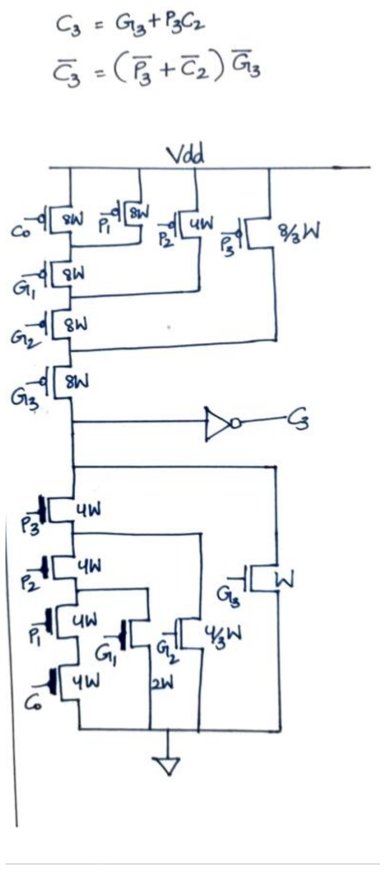


Figure: Sizing of C-3 Block

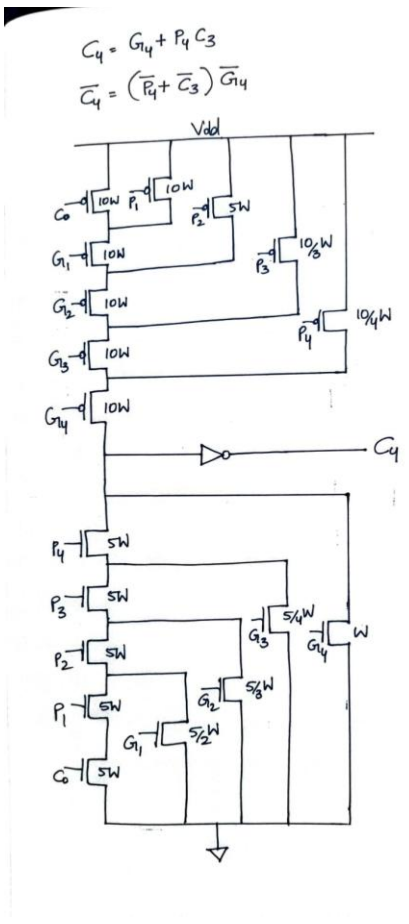


Figure: Sizing of C-4 Block

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.NGSPICE PRE-LAYOUT

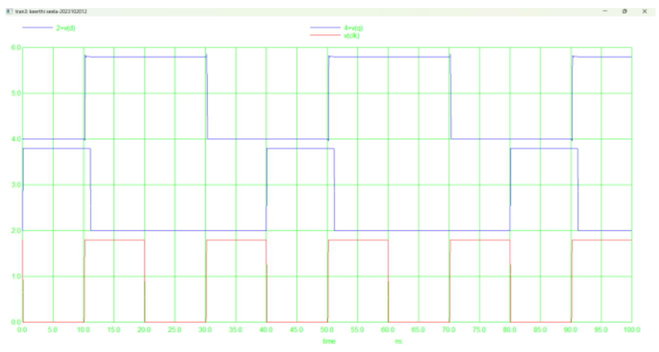


Figure: TSPC Positive Edge Flipflop

Inputs:

Vclk clk 0 pulse 1.8 0 0ns 0ns 0ns 10ns 20ns

Vd d 0 pulse 0 1.8 0ns 0ns 0ns 11ns 40ns

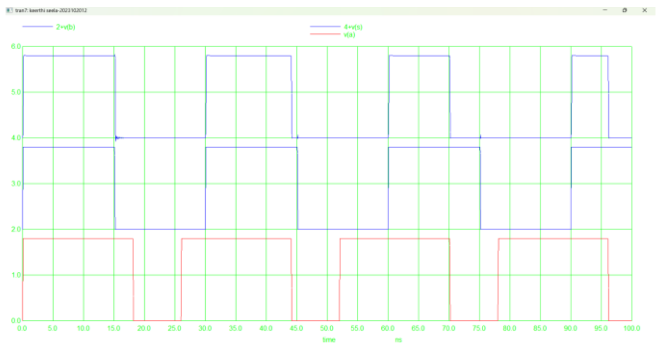


Figure: AND GATE (PTL LOGIC)

Inputs:

vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 18ns 26ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

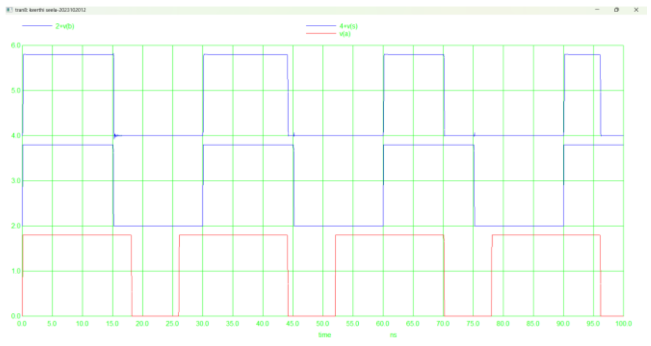


Figure: XOR GATE (PTL LOGIC)

Inputs:

vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

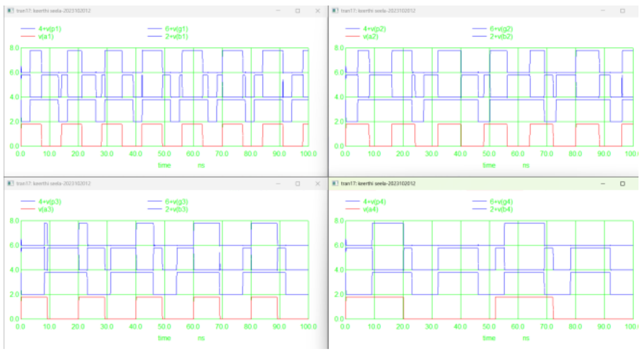


Figure: Propagate and generate block

Inputs:

vin_a1 a1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns

vin_a2 a2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns

vin_a3 a3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

vin_a4 a4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns

vin_b1 b1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns

vin_b2 b2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_b3 b3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

vin_b4 b4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns

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

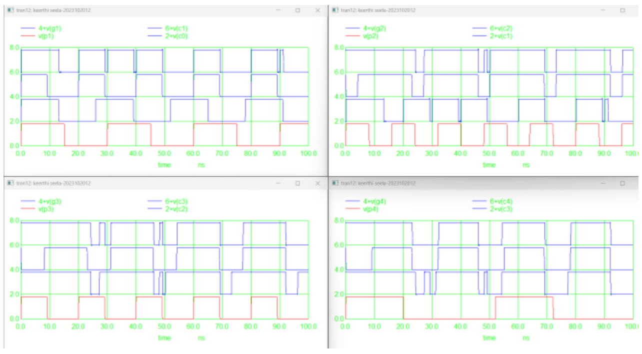


Figure: CLA Block

Inputs:

vin1 c0 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 p1 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

vin3 c1 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

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 52ns

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

vin_b2 c2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_b3 c3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

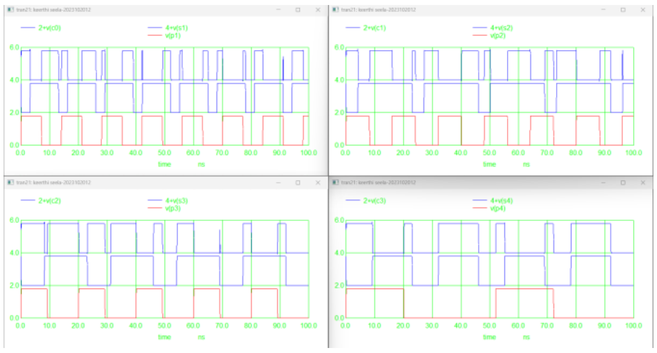


Figure: Sum Block

4. TIME CONSTRAINTS OF D-FLIPFLOP

From NGSPICE simulations, I checked upto what value before the positive edge of the clock should I not change the value of the input and this is the required setup time of D-Flipflop. The minimum amount of time that the data input (D) must be stable before the clock edge arrives. And upto what value after the positive edge of the clock should I not change the value of the input and this is the required hold time of D-Flipflop. The minimum amount of time that the data input (D) must remain stable after the clock edge has occurred. After the positive edge of clock how long does the flipflop take to give output this is the clock to Q delay of the D-Flipflop.

- **Setup Time:**
  8.7×10^{-11}s ≃0.09ns ≃0.1ns
- **Hold Time:**
  4×10^{-11}s ≃40ps ≃Negligible
- **Clock to Q Delay:**

  8.313×10^{-11}s

5. STICK DIAGRAMS

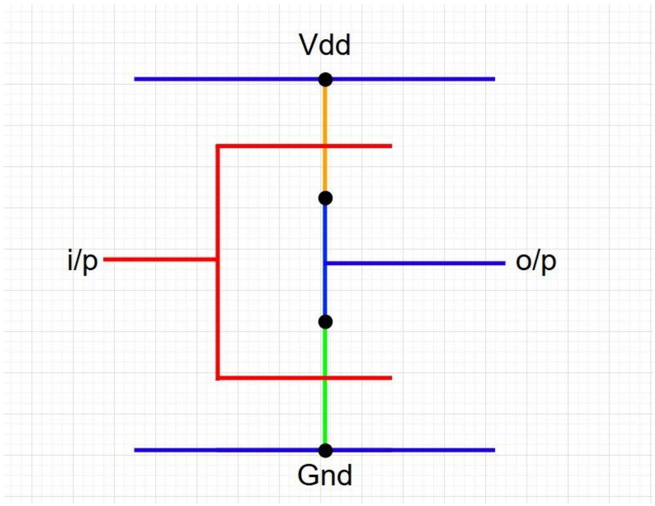


Figure 1: INVERTER

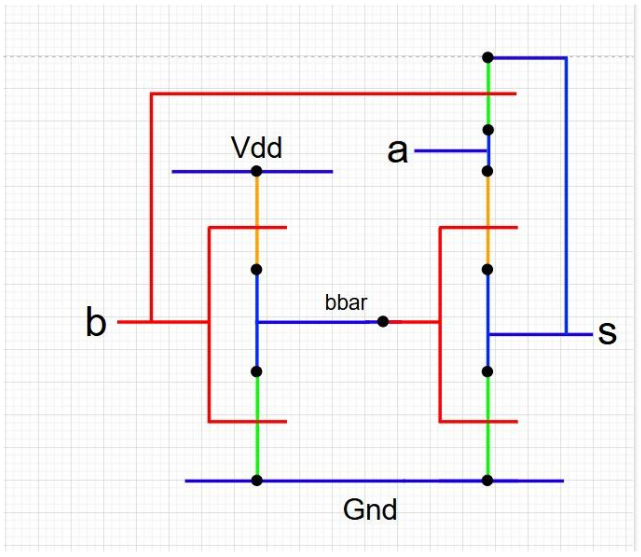


Figure 2: AND GATE

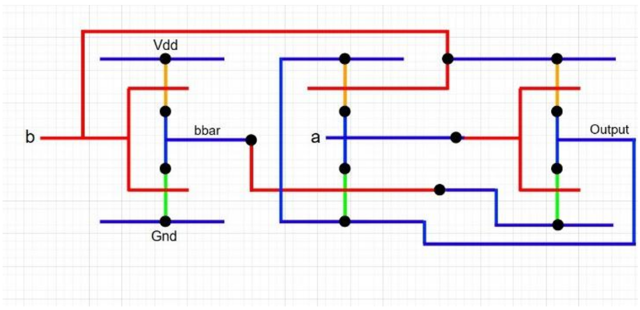


Figure 3: XOR Gate

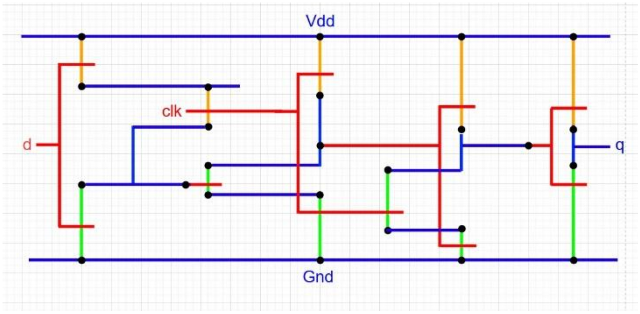


Figure 4: D-Flipflop

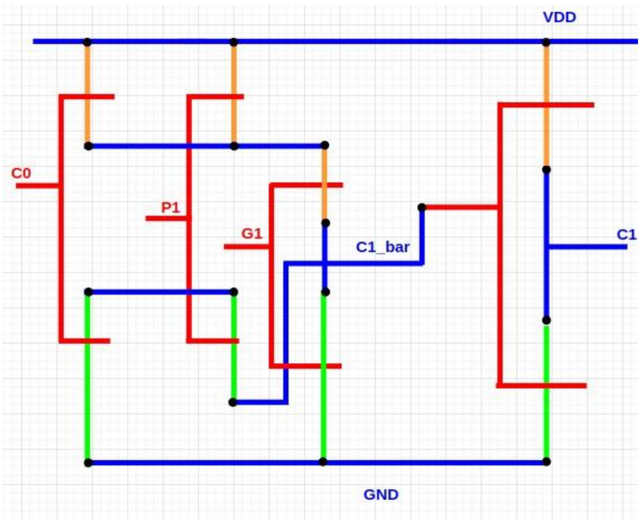


Figure 5: C-1 Block

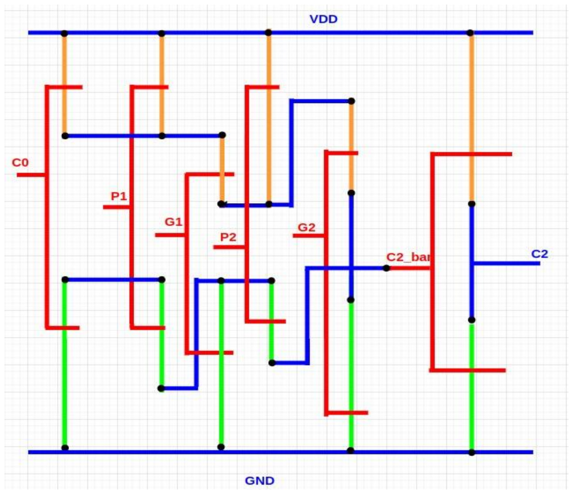


Figure 6: C-2 Block

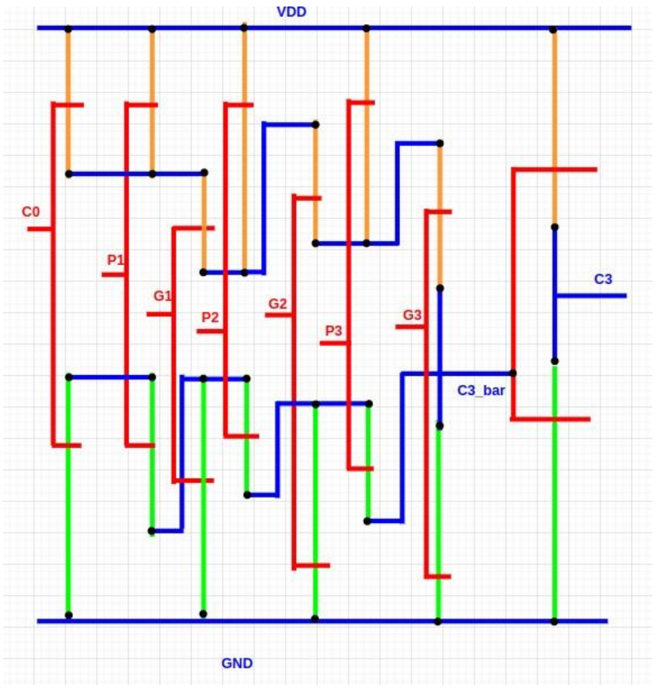


Figure 7: C-3 Block

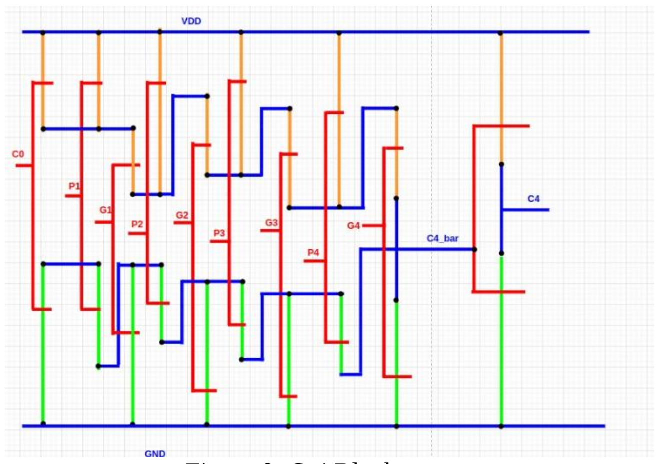


Figure 8: C-4 Block

6-I. MAGIC LAYOUT

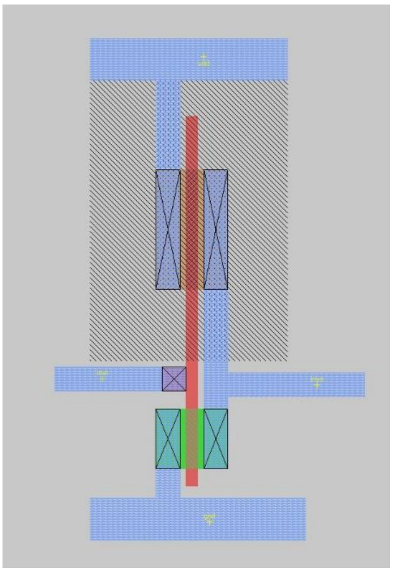


Figure: Inverter

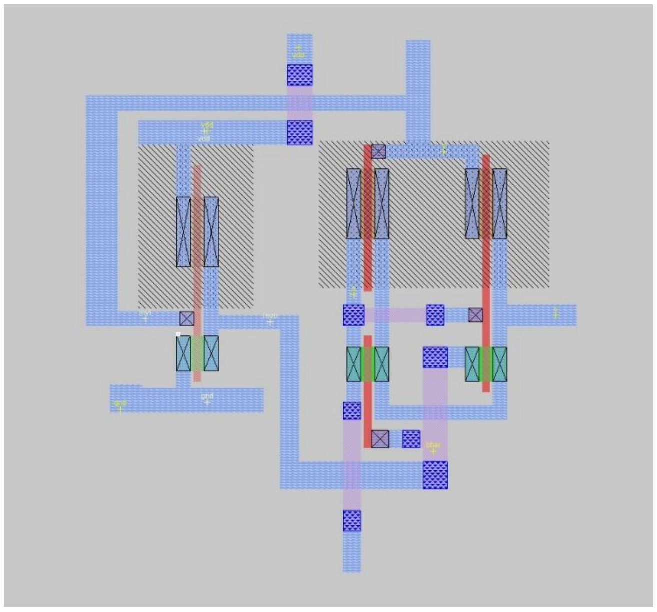


Figure: XOR GATE (PTL LOGIC)

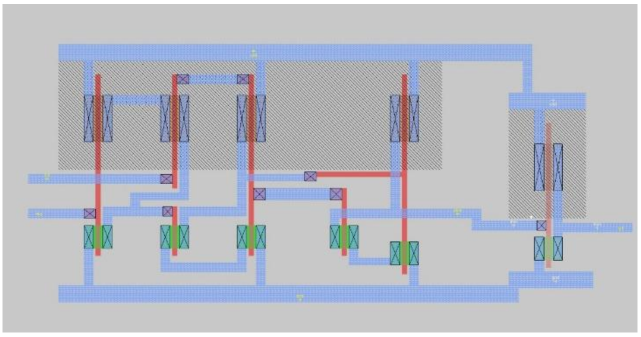


Figure: TSPC Positive Edge Flipflop

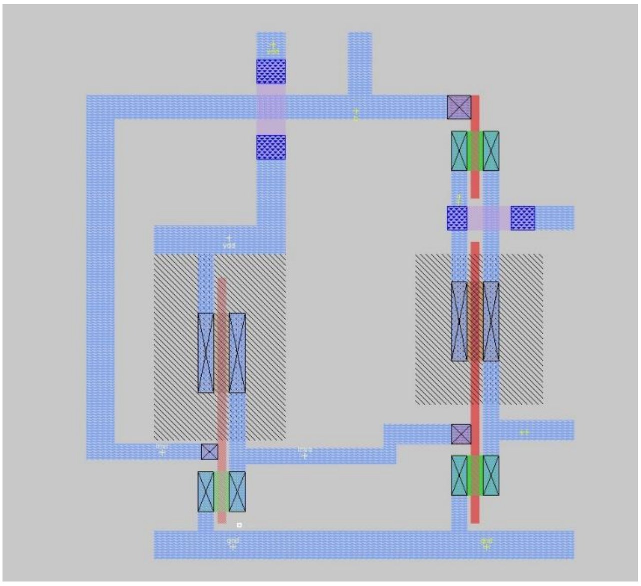


Figure: AND GATE (PTL LOGIC)

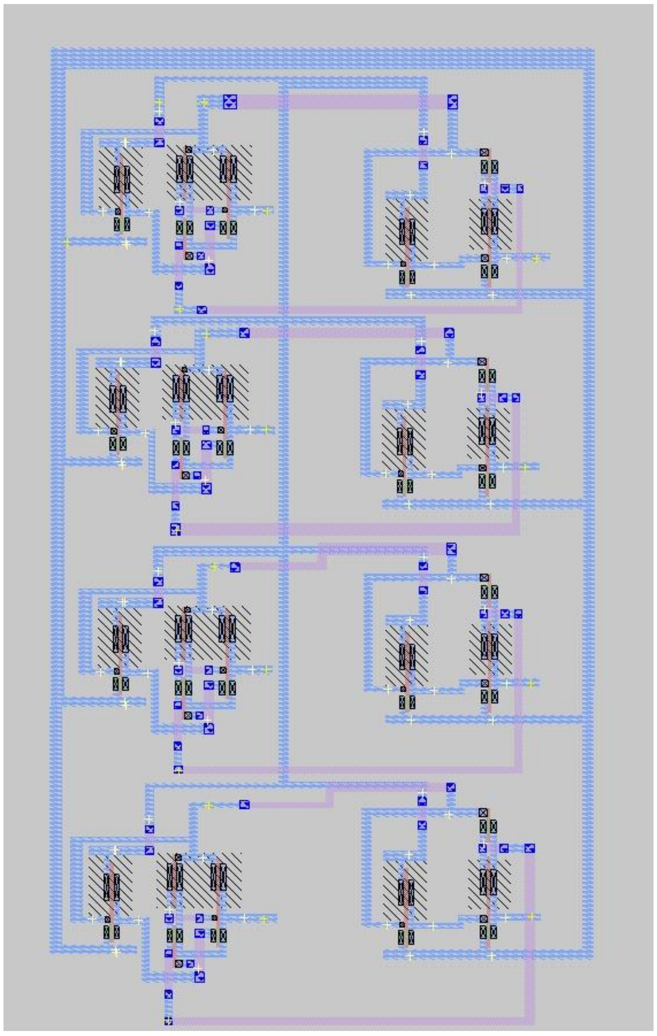


Figure: Propagate and generate block

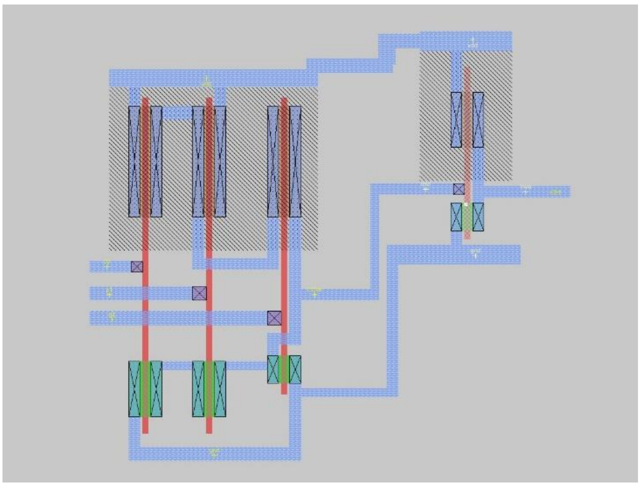


Figure: C-1 Block

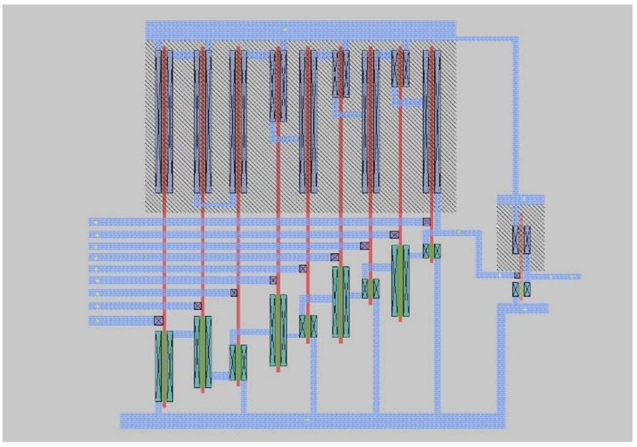


Figure: C-4 Block

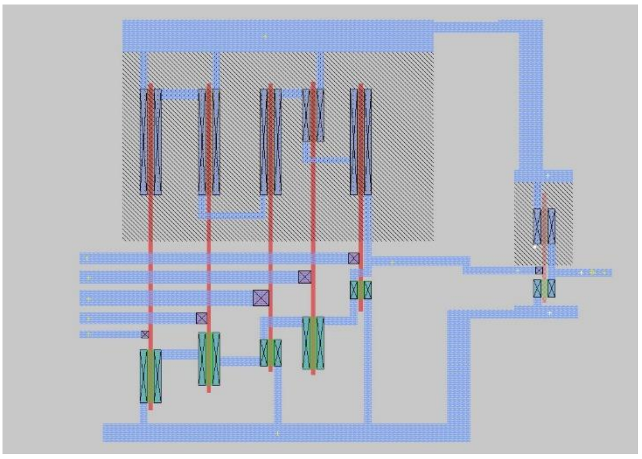


Figure: C-2 Block

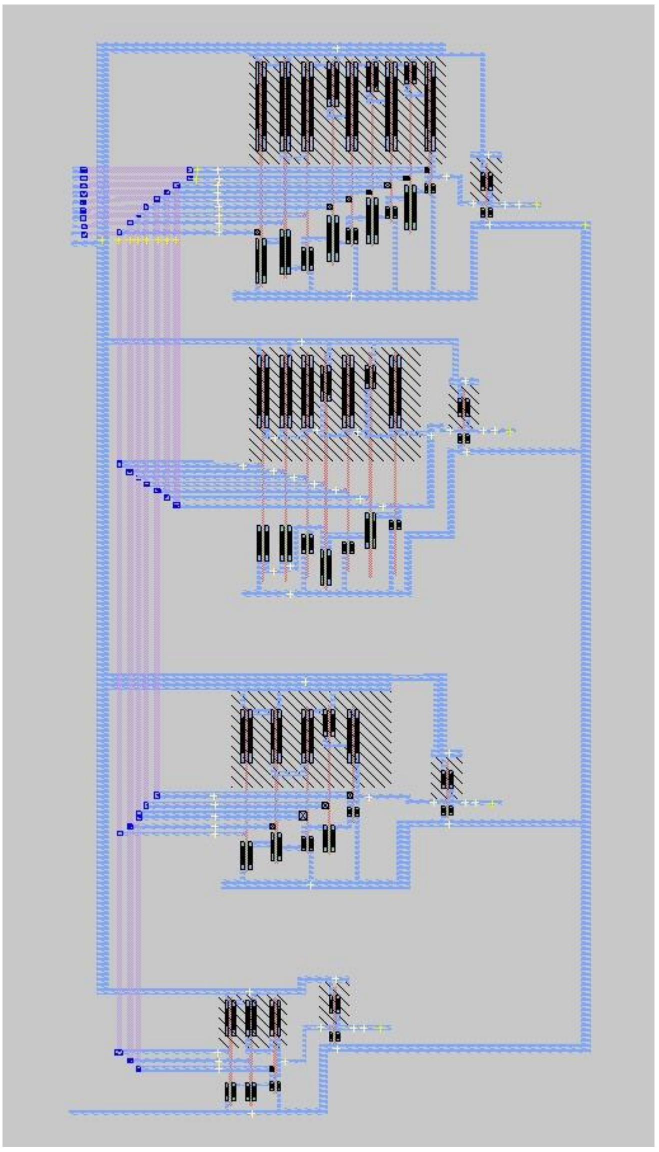


Figure: CLA Block

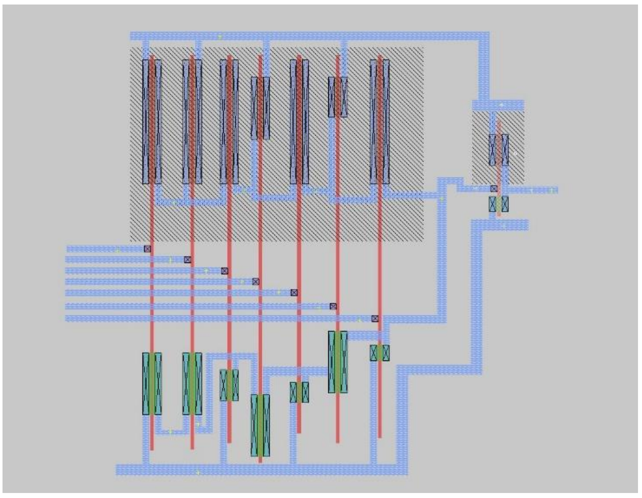


Figure: C-3 Block

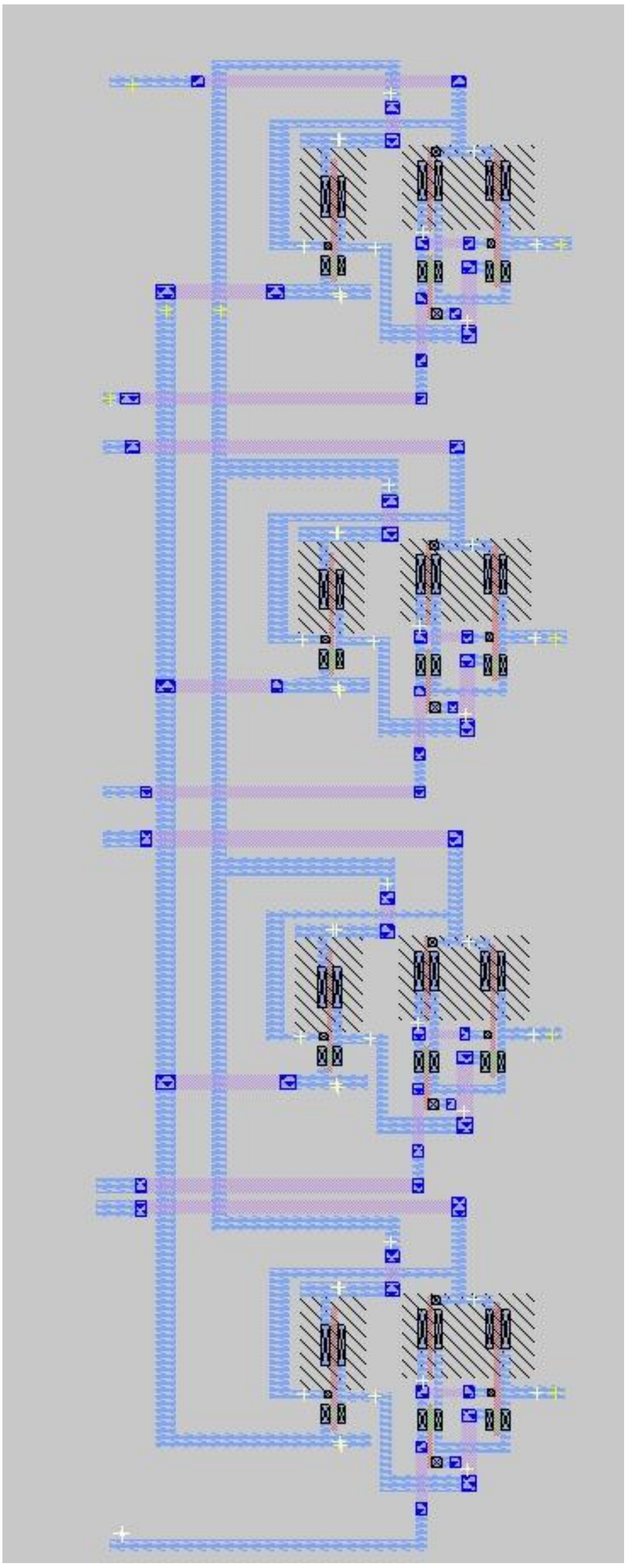


Figure: Sum Block

6-II. NGSPICE POST- LAYOUT

Inputs:

Vclk clk 0 pulse(1.8 0 0 0 0 10n 20n)

Vd d gnd pulse(0 1.8 0 0 0 15n 60n)

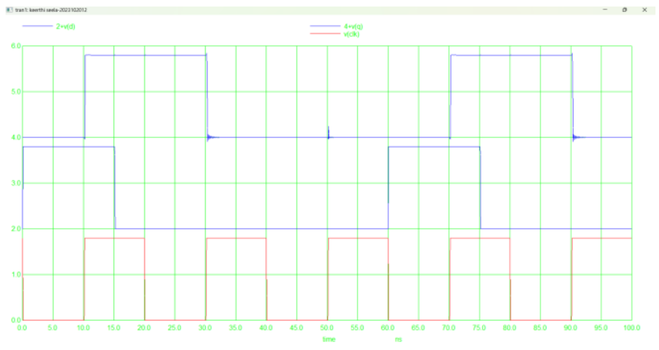


Figure: TSPC Positive Edge Flipflop

Inputs:

vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

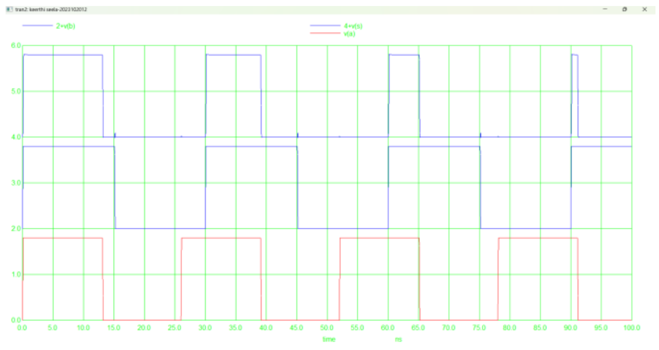


Figure: AND GATE (PTL LOGIC)

Inputs:

vin1 a 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 b 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

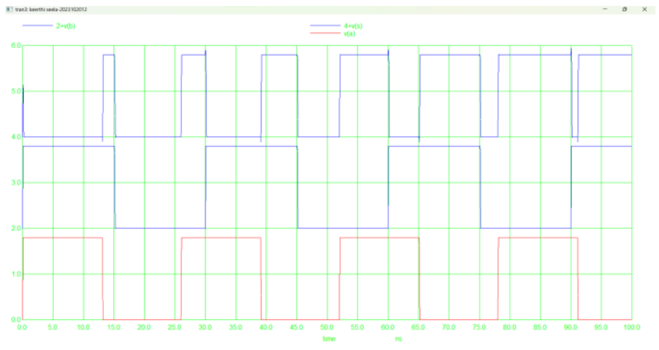


Figure: XOR GATE (PTL LOGIC)

Inputs:

vin_a1 a1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns

vin_a2 a2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns

vin_a3 a3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

vin_a4 a4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns vin_b1 b1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns

vin_b2 b2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_b3 b3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

vin_b4 b4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns

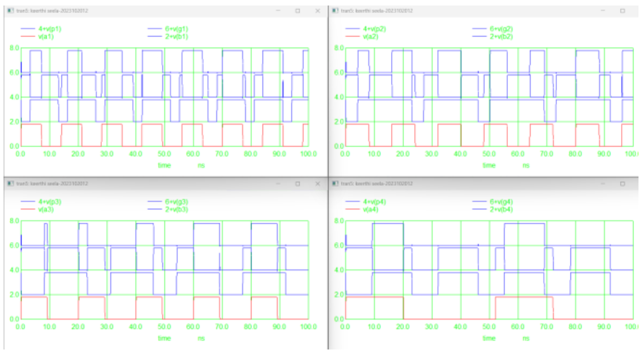


Figure: Propagate and generate block

Inputs:

vin1 c0 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 p1 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

vin3 g1 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

vin_p2 p2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns

vin_g2 g2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_p3 p3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

vin_g3 g3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

vin_p4 p4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns

vin_g4 g4 0 pulse 1.8 0 0ns 0ns 0ns 9ns 23ns

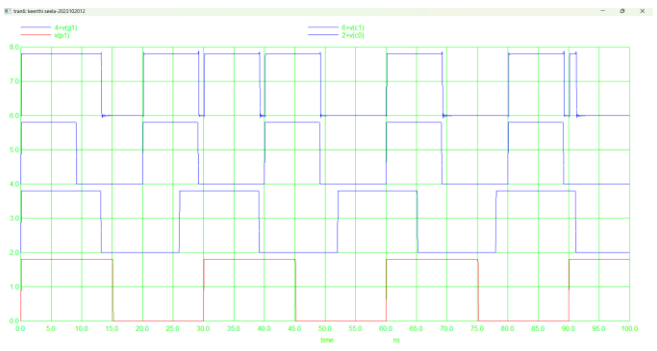


Figure: C-1 Block

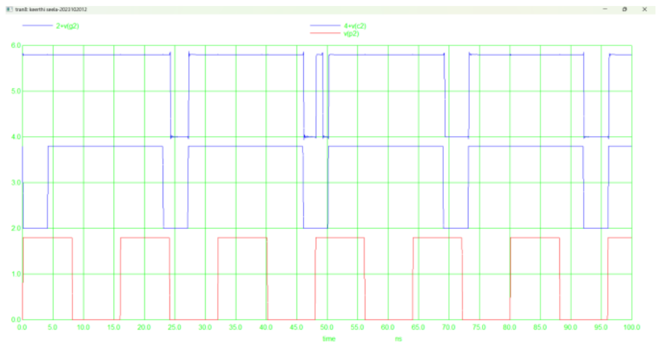


Figure: C-2 Block

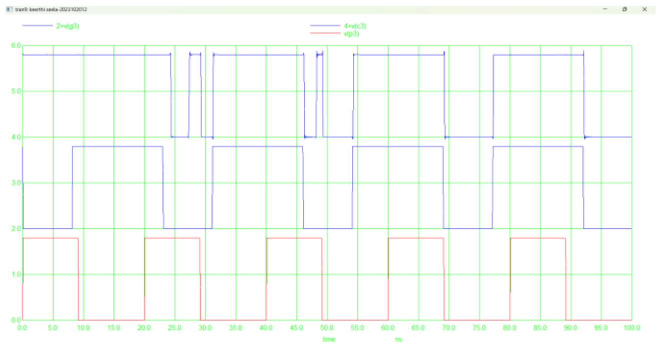


Figure: C-3 Block

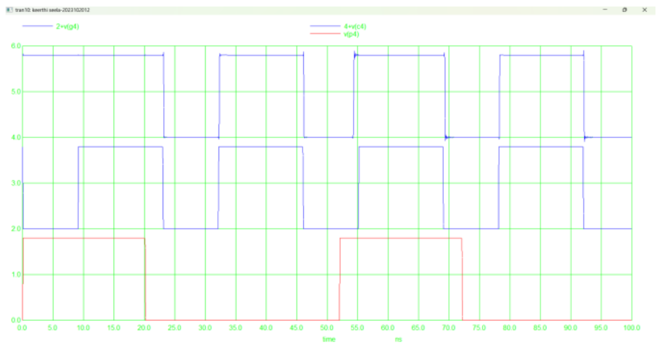


Figure: C-4 Block

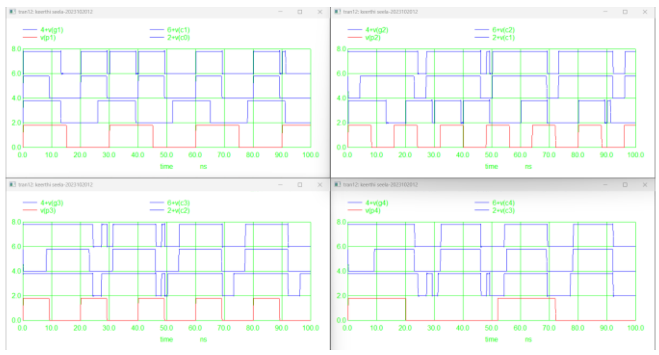


Figure: CLA Block

Inputs:

vin1 c0 0 pulse 0 1.8 0ns 0ns 0ns 13ns 26ns

vin2 p1 0 pulse 0 1.8 0ns 0ns 0ns 15ns 30ns

vin3 c1 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns

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 52ns vin_a2 p2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns

vin_b2 c2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns

vin_b3 c3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns

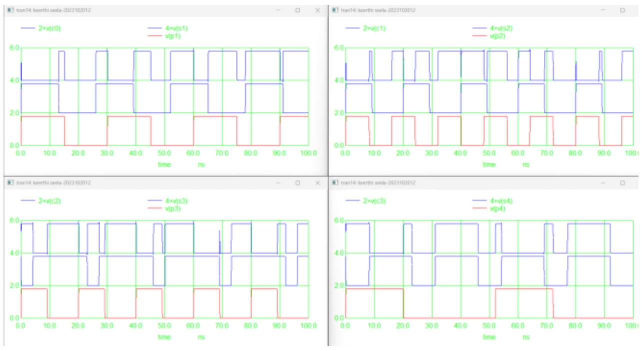
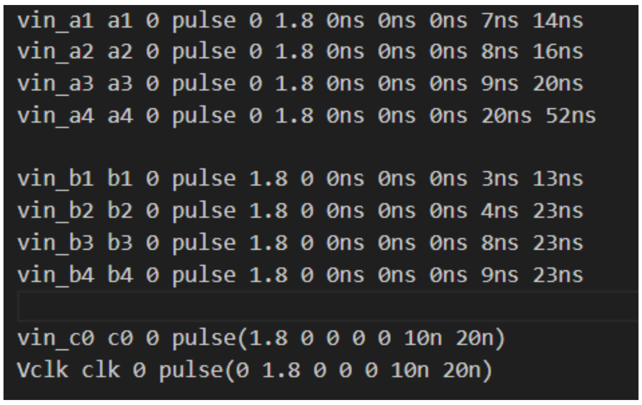


Figure: Sum Block

Post Layout we acquired parasitic Capacitances.

7. NGSPICE PRE-LAYOUT OF FULL CIRCUIT

Inputs:



```
vin_a1 a1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns
vin_a2 a2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns
vin_a3 a3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a4 a4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns

vin_b1 b1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns
vin_b2 b2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns
vin_b3 b3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns
vin_b4 b4 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)
```

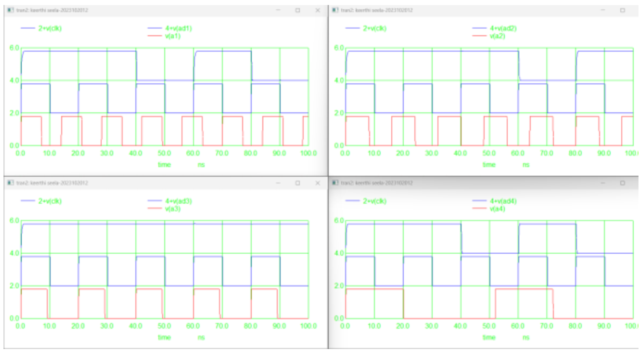


Figure: Output from Flipflop bus(a)

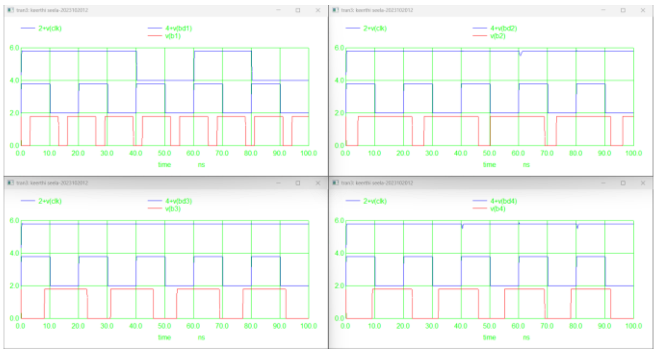


Figure: Output from Flipflop bus(b)

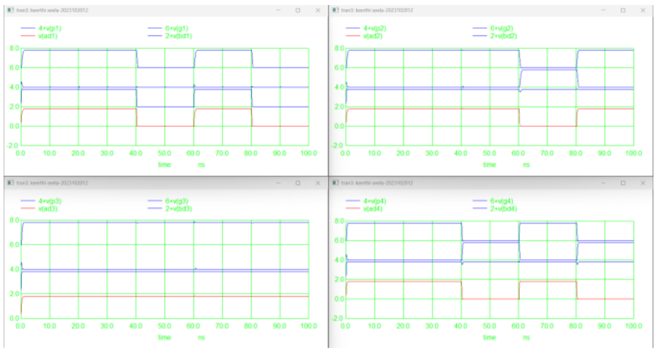


Figure: Output from PG Block

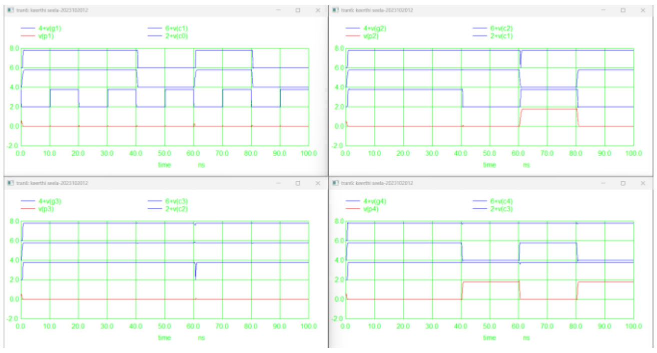


Figure: Output from CLA Block

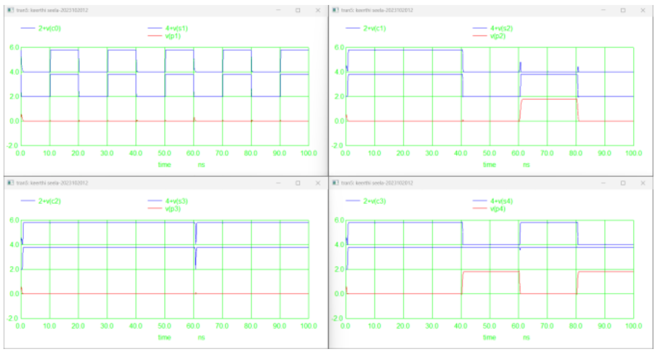


Figure: Output from Sum Block

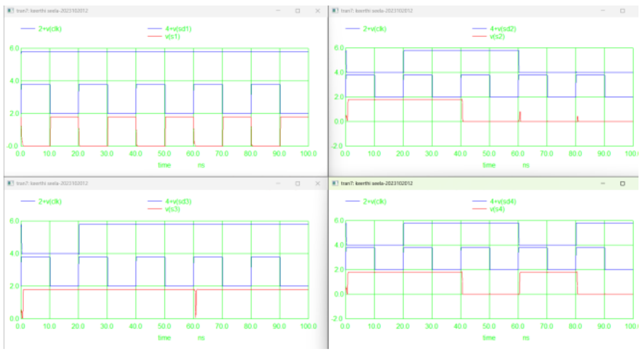


Figure: Output from Flipflop bus (Sum)

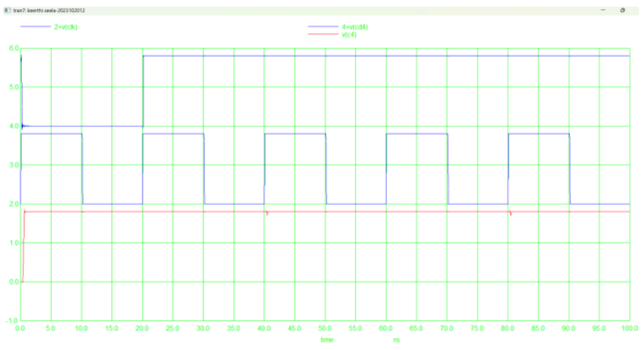


Figure: Output from Flipflop bus (Carry)

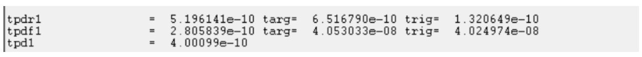

```
tpdr1                  =  5.196141e-10 targ=  6.516790e-10 trig=  1.320649e-10
tpdf1                  =  2.805839e-10 targ=  4.053033e-08 trig=  4.024974e-08
tpd1                   =  4.00099e-10
```

Figure: Worst case delay of adder

Maximum clock speed=1.75GHz
This is calculated using the formula:

$T_{clk\text{-}min}=t_{pCQ}+t_{PD}+t_{setup}$

$F_{clk\text{-}max}=1/\ T_{clk\text{-}min}$

DC Inputs:
A ($a_4\ a_3\ a_2\ a_1$) =1101
B ($b_4\ b_3\ b_2\ b_1$) =1101
Cin=0
Output Sum ($s_4\ s_3\ s_2\ s_1$) =1010
Output Carry=1

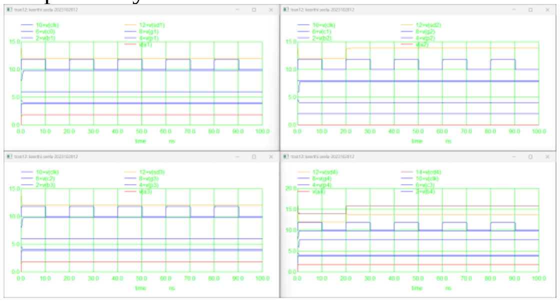


Figure: DC Output

A ($a_4\ a_3\ a_2\ a_1$) =1010
B ($b_4\ b_3\ b_2\ b_1$) =1010
Cin=0
Output Sum ($s_4\ s_3\ s_2\ s_1$) =0100
Output Carry=1

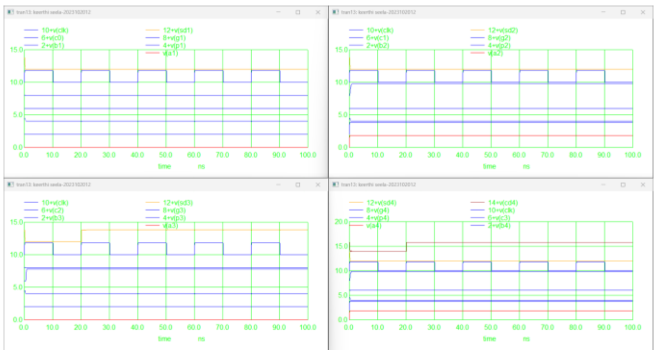


Figure: DC Output

8. FLOOR PLAN

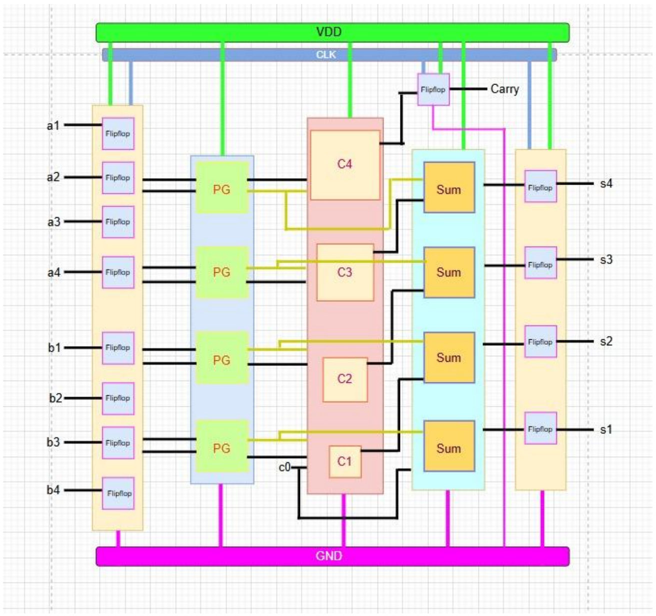


Figure: Floor Plan of Layout for complete circuit

Horizontal pitch=2586 *lambda =232.74nm

Vertical pitch=1598 *lambda =143.82nm

9. NGSPICE POST-LAYOUT

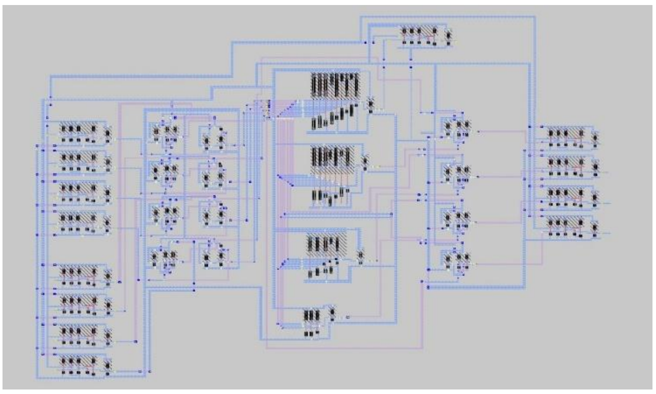
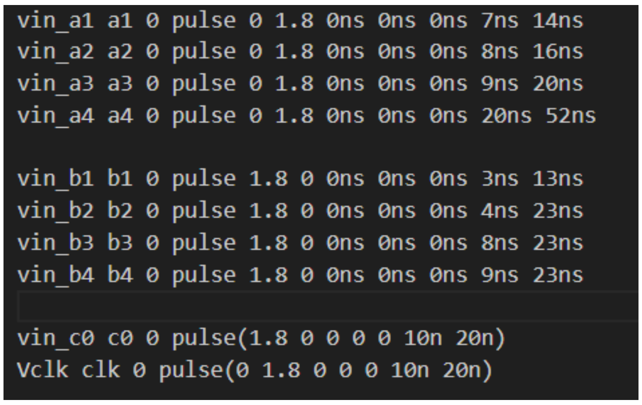


Figure: Layout for complete circuit

Inputs:



```
vin_a1 a1 0 pulse 0 1.8 0ns 0ns 0ns 7ns 14ns
vin_a2 a2 0 pulse 0 1.8 0ns 0ns 0ns 8ns 16ns
vin_a3 a3 0 pulse 0 1.8 0ns 0ns 0ns 9ns 20ns
vin_a4 a4 0 pulse 0 1.8 0ns 0ns 0ns 20ns 52ns

vin_b1 b1 0 pulse 1.8 0 0ns 0ns 0ns 3ns 13ns
vin_b2 b2 0 pulse 1.8 0 0ns 0ns 0ns 4ns 23ns
vin_b3 b3 0 pulse 1.8 0 0ns 0ns 0ns 8ns 23ns
vin_b4 b4 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)
```

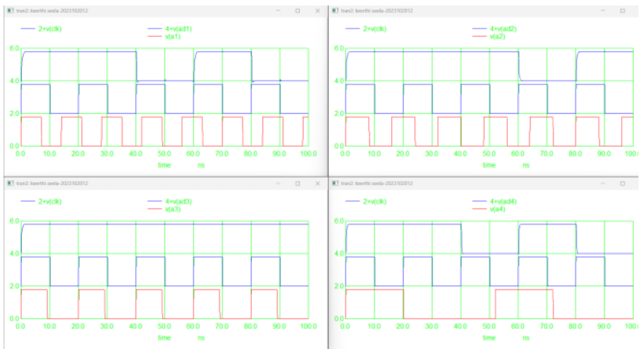


Figure: Output from Flipflop bus(a)

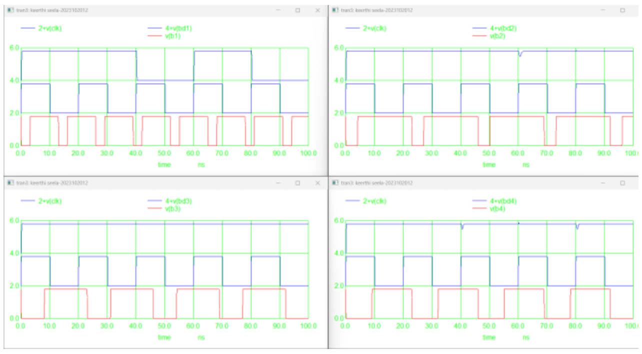


Figure: Output from Flipflop bus(b)

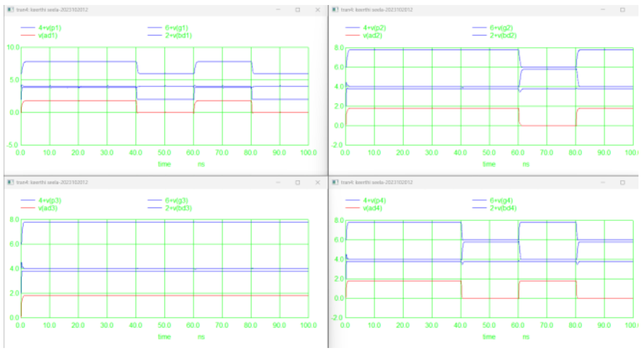


Figure: Output from PG Block

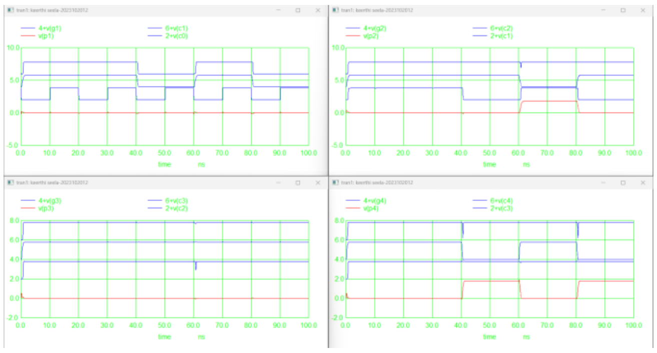


Figure: Output from CLA Block

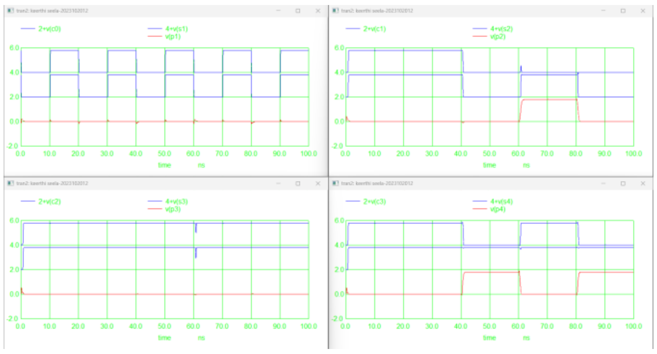


Figure: Output from Sum Block

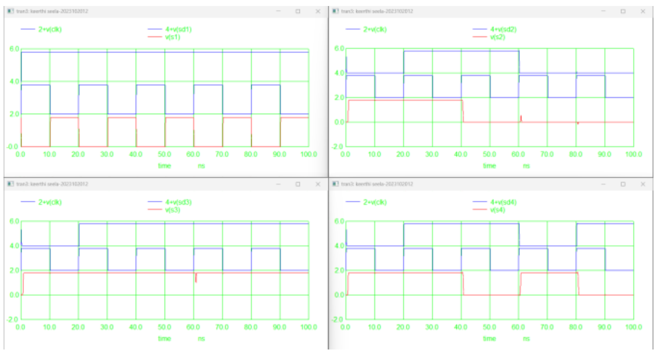


Figure: Output from Flipflop bus (Sum)

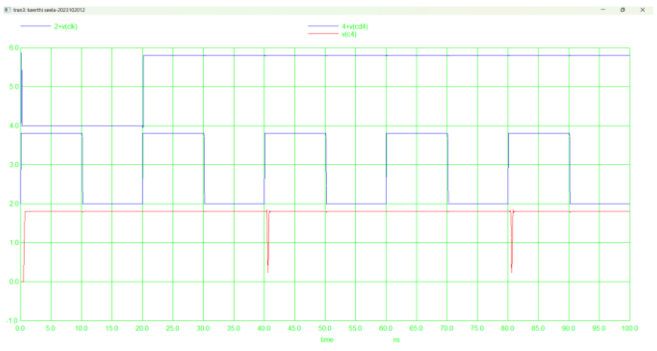


Figure: Output from Flipflop bus (Carry)

DC Inputs:

A (a_4 a_3 a_2 a_1) =1101

B (b_4 b_3 b_2 b_1) =1101

Cin=0

Output Sum (s_4 s_3 s_2 s_1) =1010

Output Carry=1

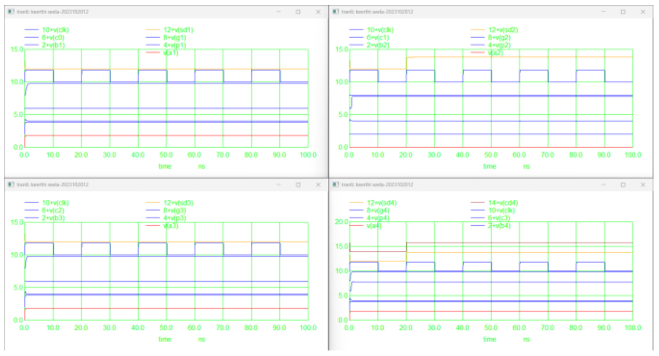


Figure: DC Output

A (a_4 a_3 a_2 a_1) =1010
B (b_4 b_3 b_2 b_1) =1010
Cin=0
Output Sum (s_4 s_3 s_2 s_1) =0100
Output Carry=1

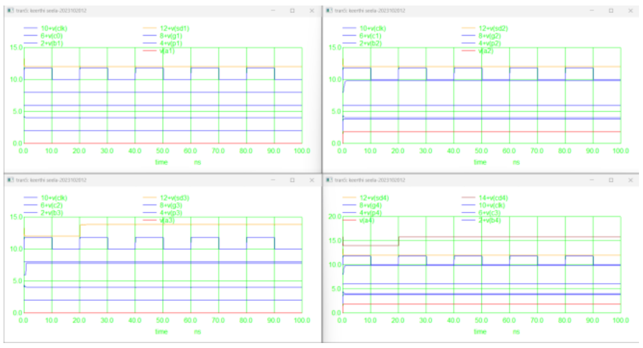


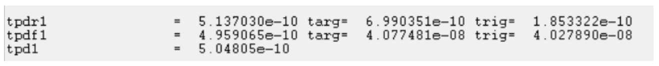
Figure: DC Output

	Pre-Layout	Post-Layout
Setup time	8.7×10^{-11}s	9×10^{-11} s
Hold time	4×10^{-11} s	4.6×10^{-11} s
Clock to Q Delay	8.313×10^{-11}	8.952×10^{-11} s
Worst Case Delay of Adder	40×10^{-11} s	50.4×10^{-11} s
Minimum Time period	57.013×10^{-11} s	68.3×10^{-11} s
Maximum clock frequency	1.75GHz	1.47GHz

10. TIME CONSTRAINTS FOR ADDER POST-LAYOUT

D-Flipflop time constraints:

- **Setup Time:**
  9×10^{-11}s =0.09ns ≃0.1ns
- **Hold Time:**
  4.6×10^{-11}s ≃46ps ≃Negligible
- **Clock to Q Delay:**
  8.952×10^{-11}s



```
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
```

Figure: Delay of adder

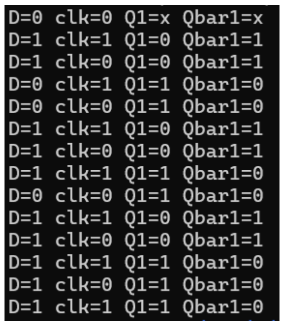
Maximum clock frequency=1.47GHz

This is calculated using the formula:

$T_{clk\text{-}min}=t_{pCQ}+t_{PD}+t_{setup}$

$F_{clk\text{-}max}=1/\ T_{clk\text{-}min}$

11. VERILOG HDL



```
D=0 clk=0 Q1=x Qbar1=x
D=1 clk=1 Q1=0 Qbar1=1
D=1 clk=0 Q1=0 Qbar1=1
D=0 clk=1 Q1=1 Qbar1=0
D=0 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=0 Qbar1=1
D=1 clk=0 Q1=0 Qbar1=1
D=1 clk=1 Q1=1 Qbar1=0
D=0 clk=0 Q1=1 Qbar1=0
D=1 clk=1 Q1=0 Qbar1=1
D=1 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
```

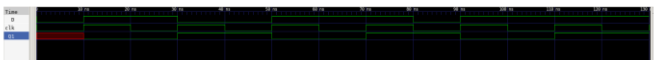
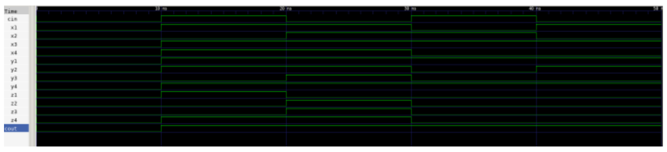
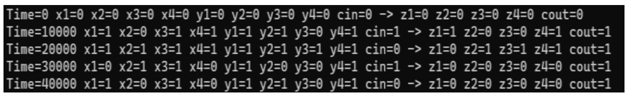


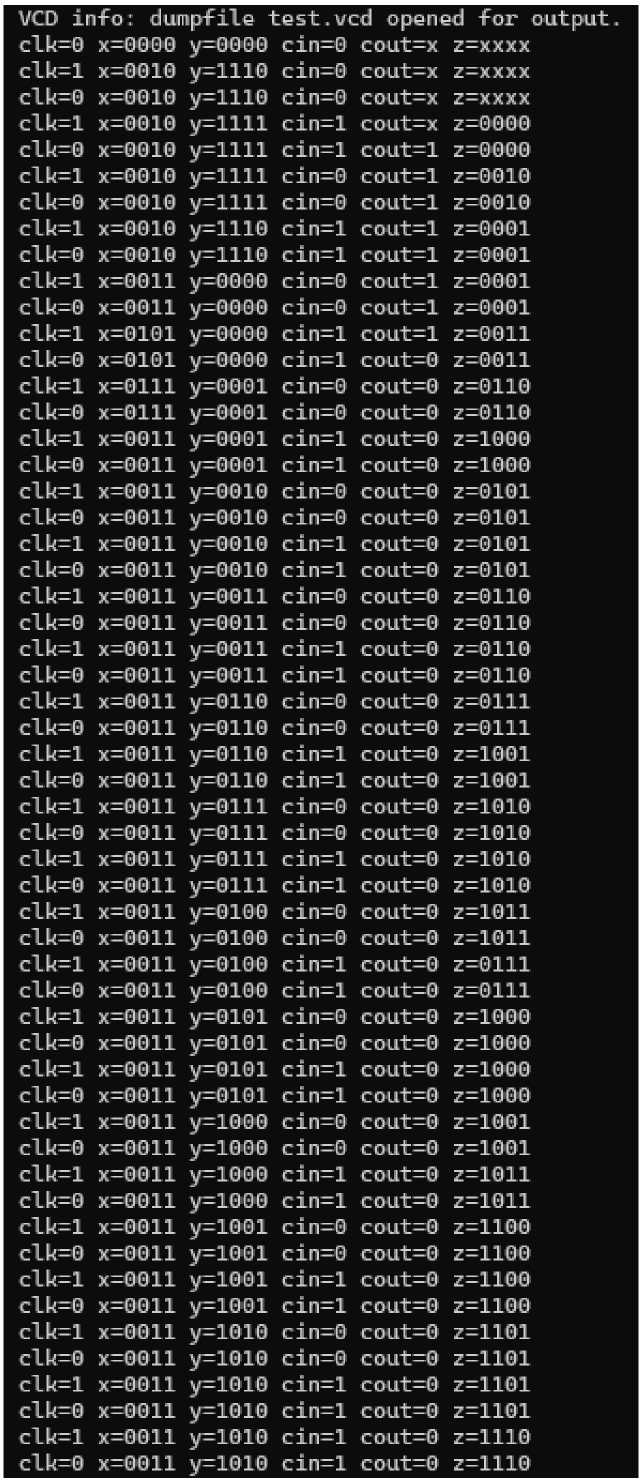
Figure: Output of D-Flipflop





```
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
```

Figure: Output of CLA Block



```
VCD info: dumpfile test.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=1111 cin=1 cout=x z=0000
clk=0 x=0010 y=1111 cin=1 cout=1 z=0000
clk=1 x=0010 y=1111 cin=0 cout=1 z=0010
clk=0 x=0010 y=1111 cin=0 cout=1 z=0010
clk=1 x=0010 y=1110 cin=1 cout=1 z=0001
clk=0 x=0010 y=1110 cin=1 cout=1 z=0001
clk=1 x=0011 y=0000 cin=0 cout=1 z=0001
clk=0 x=0011 y=0000 cin=0 cout=1 z=0001
clk=1 x=0101 y=0000 cin=1 cout=1 z=0011
clk=0 x=0101 y=0000 cin=1 cout=0 z=0011
clk=1 x=0111 y=0001 cin=0 cout=0 z=0110
clk=0 x=0111 y=0001 cin=0 cout=0 z=0110
clk=1 x=0011 y=0001 cin=1 cout=0 z=1000
clk=0 x=0011 y=0001 cin=1 cout=0 z=1000
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=0110 cin=0 cout=0 z=0111
clk=0 x=0011 y=0110 cin=0 cout=0 z=0111
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=0100 cin=0 cout=0 z=1011
clk=0 x=0011 y=0100 cin=0 cout=0 z=1011
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=1000 cin=0 cout=0 z=1001
clk=0 x=0011 y=1000 cin=0 cout=0 z=1001
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
clk=1 x=0011 y=1001 cin=1 cout=0 z=1100
clk=0 x=0011 y=1001 cin=1 cout=0 z=1100
clk=1 x=0011 y=1010 cin=0 cout=0 z=1101
clk=0 x=0011 y=1010 cin=0 cout=0 z=1101
clk=1 x=0011 y=1010 cin=1 cout=0 z=1101
clk=0 x=0011 y=1010 cin=1 cout=0 z=1101
clk=1 x=0011 y=1010 cin=1 cout=0 z=1110
clk=0 x=0011 y=1010 cin=1 cout=0 z=1110
```

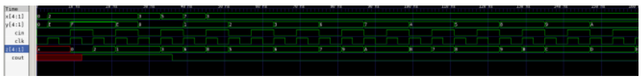
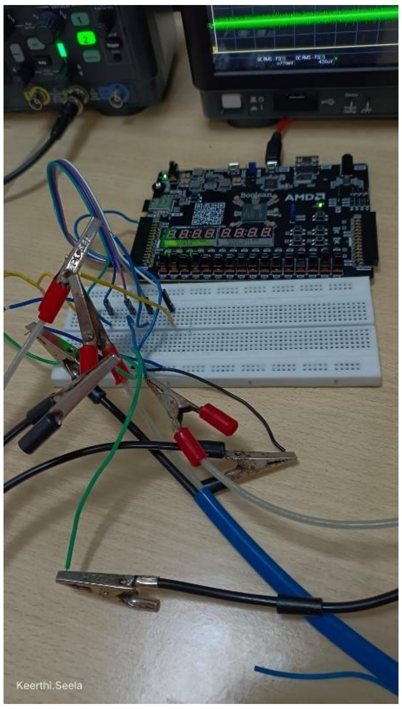
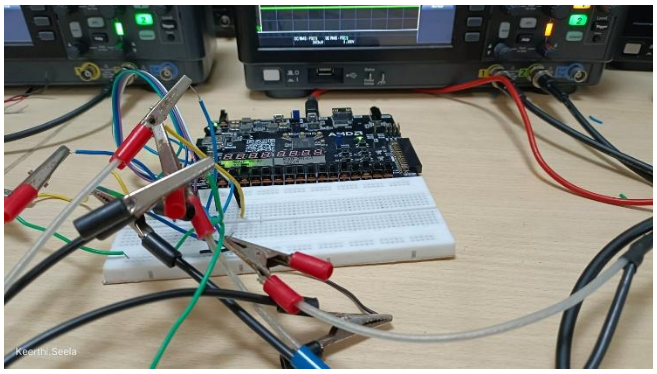


Figure: Final Output

12. FPGA





Inputs:

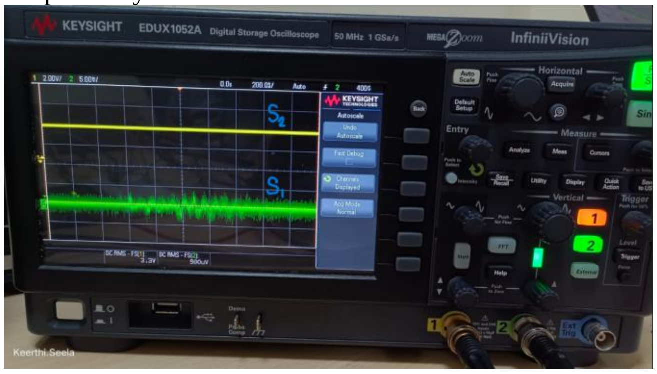
A (a_4 a_3 a_2 a_1) =1101

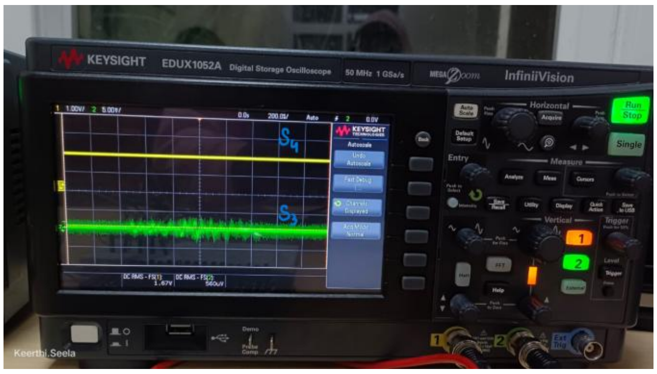
B (b_4 b_3 b_2 b_1) =1101

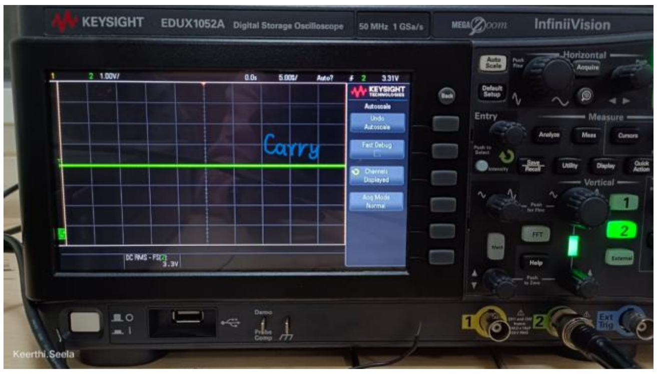
Cin=0

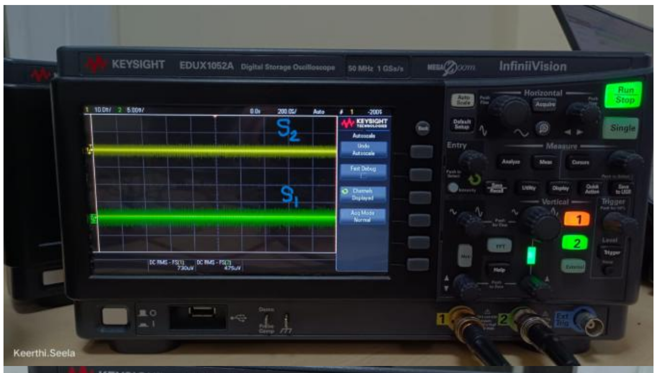
Output Sum (s_4 s_3 s_2 s_1) =1010

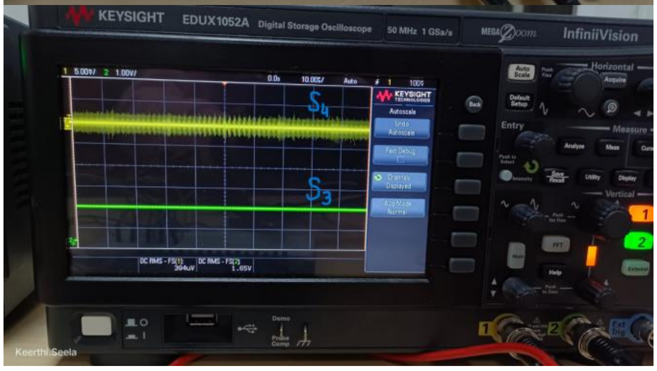
Output Carry=1











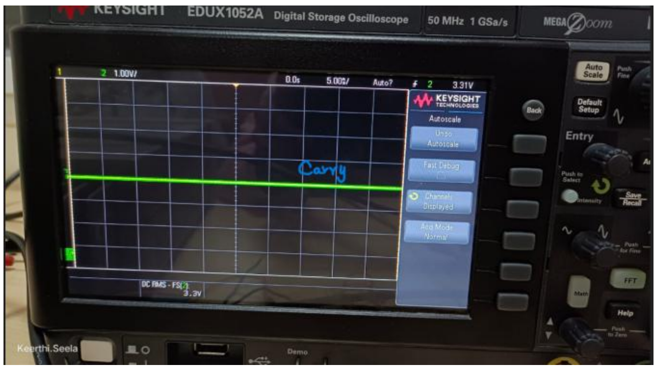


Figure: Final Output

A (a_4 a_3 a_2 a_1) =1010

B (b_4 b_3 b_2 b_1) =1010

Cin=0

Output Sum (s_4 s_3 s_2 s_1) =0100

Output Carry=1

Figure: Final Output