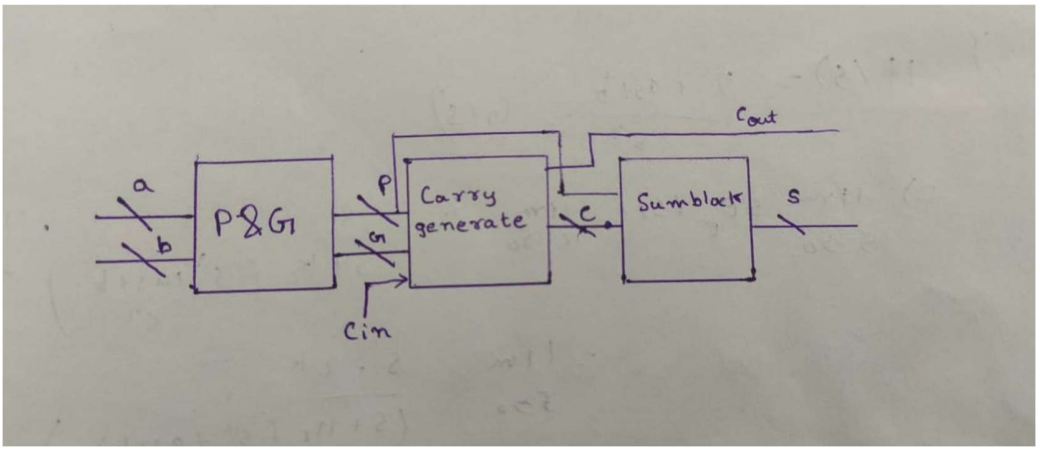
VLSI project report

Question-1

Why carry lookahead adder??

Our primary aim is to add two 4-bit numbers. This can be implemented in many ways. For instance, we can use ripple carry adder. But, there is huge delay involved as each carry is calculated from the previous carry. So one of the best implementation known to minimise the time delay(the time elapsed between for the output to appear after giving the input) is by using a carry lookahead adder.

Structure of carry lookahead adder:



We have 3 blocks in our structure. We take the two 4-bit numbers a and b. Then we pass it through P and G block. The values are then passed into the Carry generate block whose values are passed into the sum block to obtain the Sum.

$$P_i = A_i \oplus B_i \qquad S_i = P_i \oplus C_i$$
$$G_i = A_i B_i \qquad C_{i+1} = G_i + P_i C_i$$

The following operations are done in the blocks using AND, OR, XOR gates.

Adding D-flipflops at input and output:

We have a problem here. Let us say we want to compute the sum for 10 different values, one after the other and these are given at the input as a stream. If they are sent with a very high frequency, then there is a chance that the time period will be lower than the computational delay. This will result in wrong values at the output because values will be changed before the computation resulting in the computation of wrong values.

To prevent this, we add D-flip flops so that the values are sent at the rising edge of the clock. By doing so, we are blocking the next input combination until the completion of the clock cycle ensuring that there is no computational error. If we carefully set the maximum clock frequency based on the delay of the carry lookahead adder block, then we get reliable and accurate outputs at the next rising edge of the clock. Simultaneously, at the next rising edge, the next input combinations are sent into the cla block.

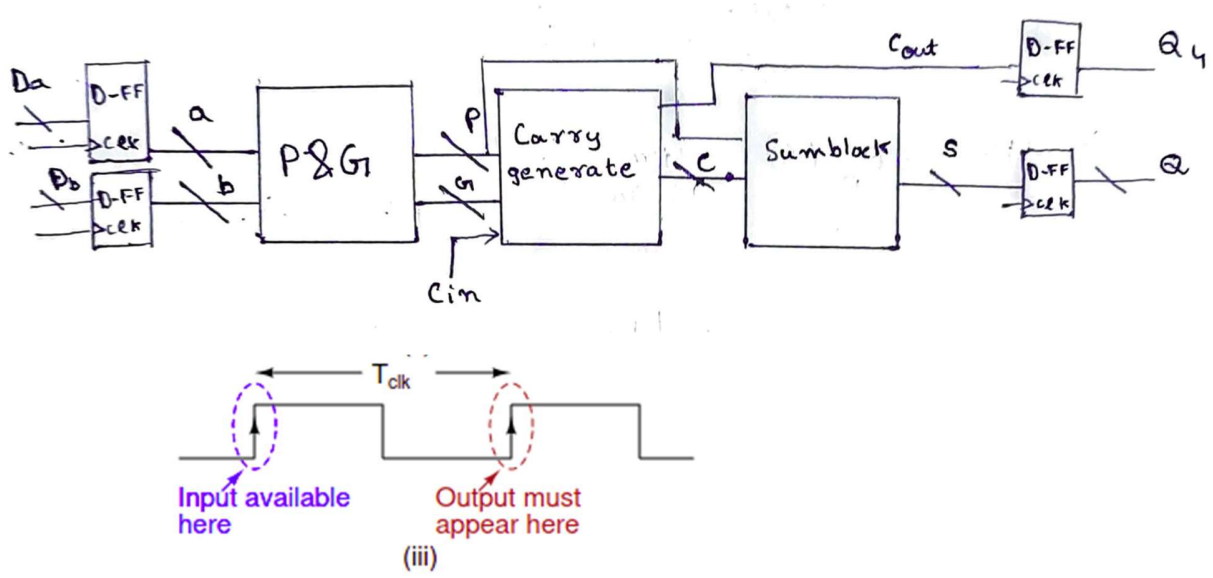


Figure 1

So in short, the components of my structure are:

1) D-flipflops : To provide sequencing

2) P&G: Calculates the Xor and And of the input bits to given P and G respectively

3) Carry Generate: Calculates the Carry

4) Sum block: Calculates the Xor of P and Carry to give the resultant sum.

Question-2, 3

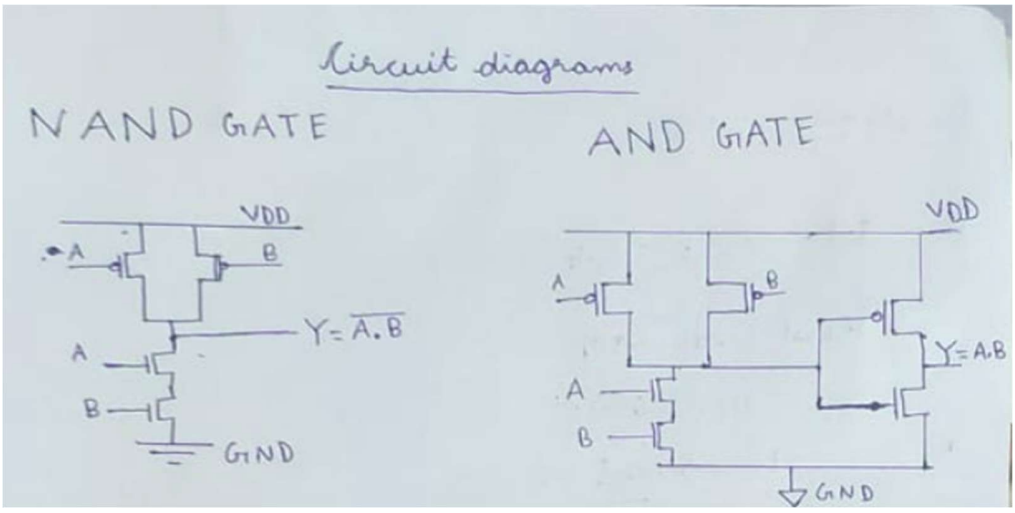
Logic style: The logic style that I have used is the **complementary CMOS logic** where we have a pull up network(consisting of PMOS) and pull down network(consisting of NMOS). At any point of time, the output is driven strongly by Vdd or Gnd. So it is also called as static CMOS logic.

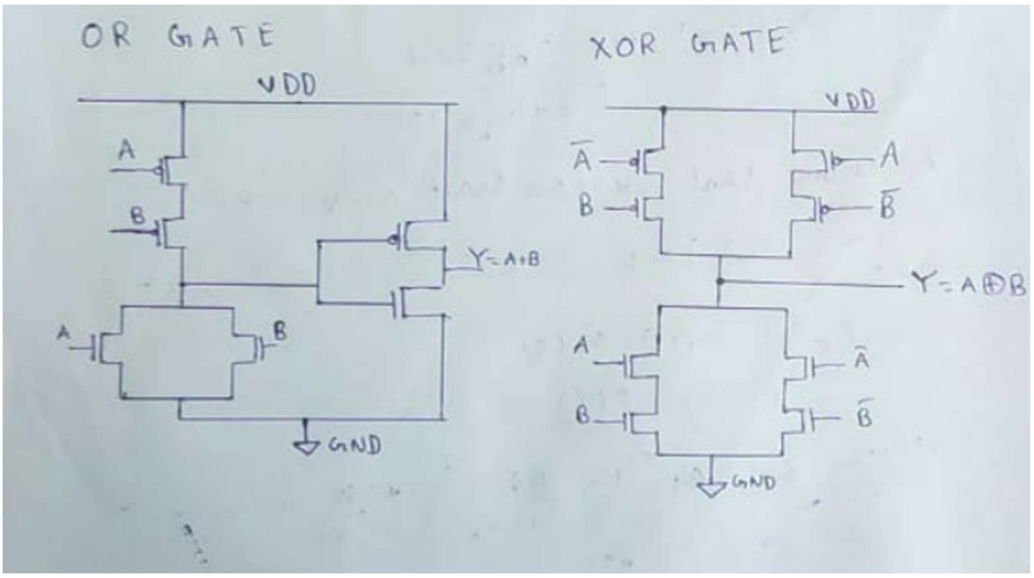
Reason for selecting the logic:

We want to design an adder. It involves a lot of calculations with 0 and 1. If I take a logic where we have bad high values(I.e. high value <1.8v), due to the many calculations involving 1s, there maybe a point where the high drops to 0.6-0.8v and thus it may appear as low.

To avoid this problem, I have taken CMOS static logic where we have **good high and low values**. Every good thing comes with a trade off. Here the **trade off** is **area** as the number of transistors required is a lot more when compared to logics like pass transistor logic which have bad highs.

Before discussing the details and topology of the blocks, let us look at the circuit diagrams of the gates for better clarity.





For each of the gates, we have pmos and nmos.

PMOS- Width=20λ Length=2λ

NMOS- Width=10λ Length=2λ

Netlist

We have implemented the circuit in ngspice. The following are the subcircuit snippets

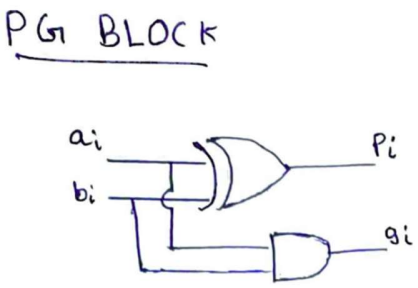
```
.subckt and a b y
.param width_N={width}
.param width_P={2*width}

M1 N 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 N 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 N a D1 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 D1 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}

M5 y N 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 y N 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}
.ends and
```

```
.subckt nand a b y
.param width_N={width}
.param width_P={2*width}

M1 y 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 y 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 y a D1 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 D1 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}

.ends nand
```

```
.subckt or a b y
.param width_N={width}
.param width_P={2*width}

M1 D1 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 N b D1 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 N 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 N 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}

M5 y N 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 y N 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}
.ends or
```

```
.subckt xor a b y
.param width_N={width}
.param width_P={2*width}
M1 a_bar 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}
M2 a_bar 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}

M3 b_bar 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}
M4 b_bar 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}


M5 D1 b_bar 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 y a D1 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 D2 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}

M8 y a_bar D2 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}

M9 y b D3 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 D3 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}

M11 y b_bar D4 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}

M12 D4 a_bar 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}
```

P and G block:



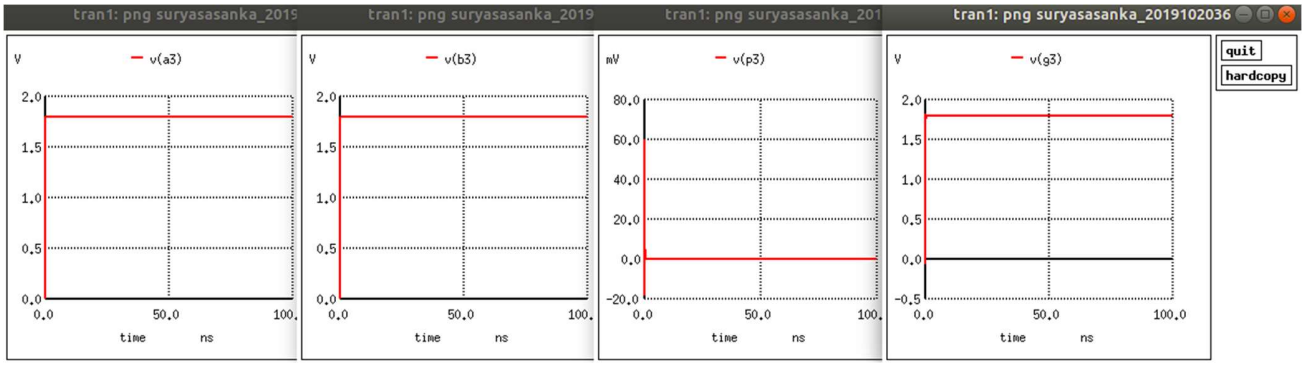
This is the basic topology of PG-block. We simply extend the idea to 4-bits.

Ngspice netlist:

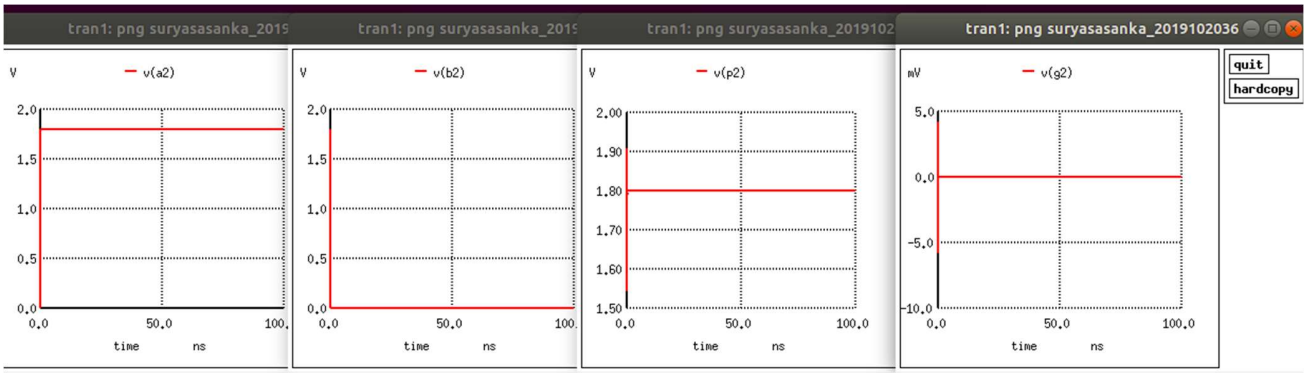
```
*---------------------------------P and G block
x1 A0 B0 P0 xor
x2 A1 B1 P1 xor
x3 A2 B2 P2 xor
x4 A3 B3 P3 xor

x5 A0 B0 G0 and
x6 A1 B1 G1 and
x7 A2 B2 G2 and
x8 A3 B3 G3 and
```

The netlist is self-explanatory given the fact that we already have written subcircuits.

Plots:

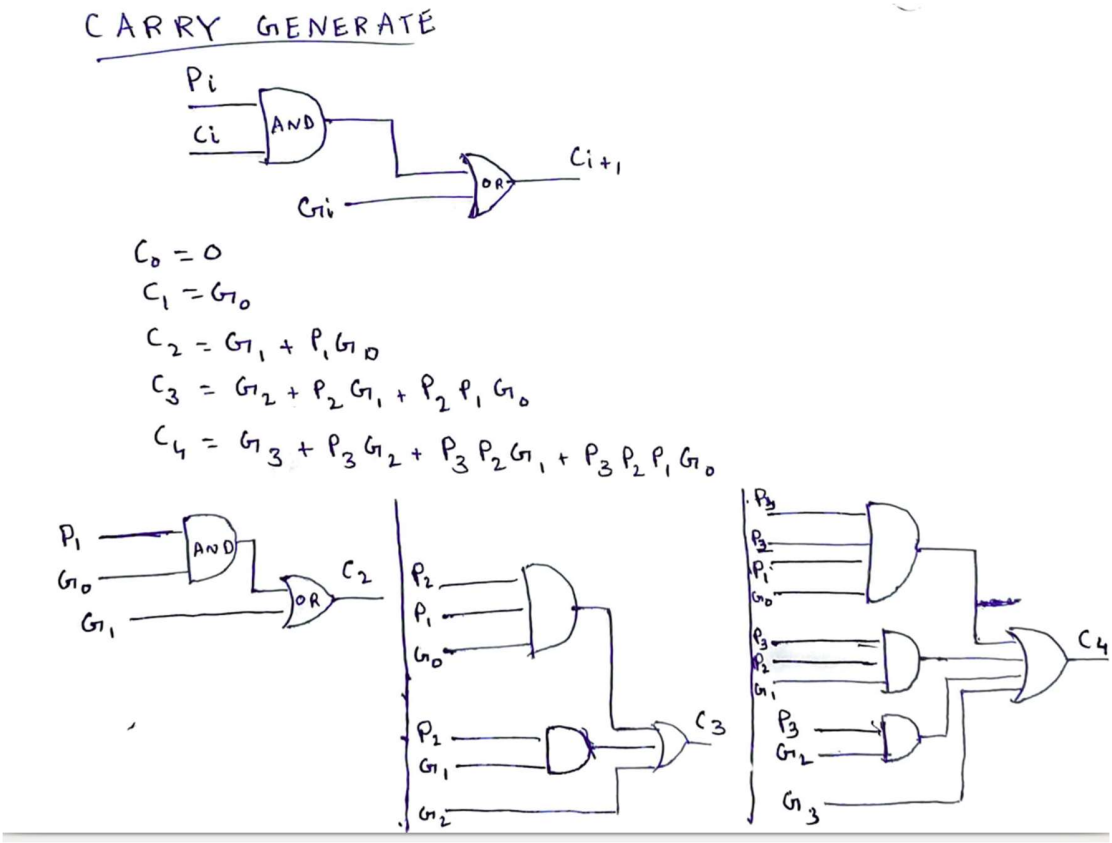


A3=1, B3=1, P3=0, G3=1



A2=1, B2=0, P2=1, G2=0

Carry generate block:

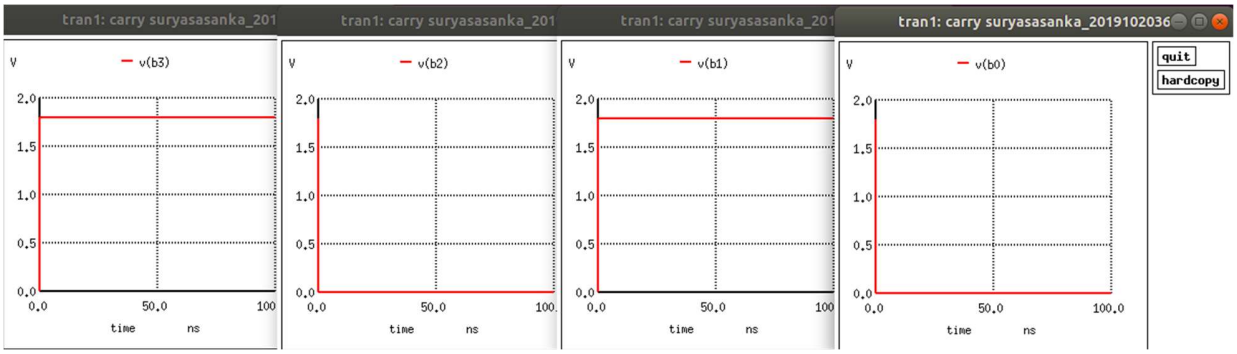


This is the topology of Carry generate block. We take the input bit C0 to be 0. Then we compute the rest of the carry values based on the above circuit.

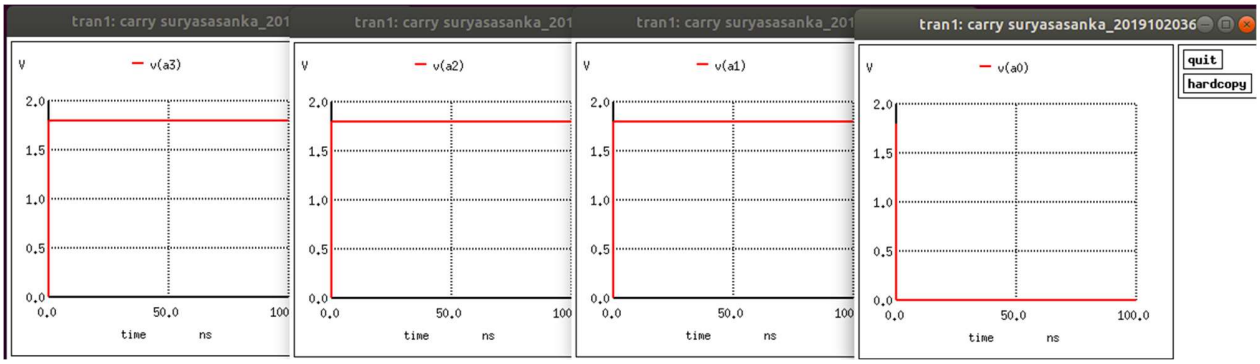
Ngspice netlist:

```
*----------------------------CLA block

V_c1 C1 G0 '0'            $G0

x9 P1 G0 PG10 and         $P1G0
x10 G1 PG10 C2 or         $G1+P1G0

x11 P2 PG10 PPG210 and    $P2P1G0
x12 P2 G1 PG21 and        $P2G1
x13 PG21 PPG210 PGPPG or  $P2G1+P2P1G0
x14 G2 PGPPG C3 or        $C3=G2+P2G1+P2P1G0

x15 P3 PPG210 PPPG3210 and    $P3P2P1G0
x16 P3 PG21 PPG321 and        $P3P2G1
x17 PPPG3210 PPG321 last1 or  $P3P2G1+P3P2P1G0

x18 P3 G2 PG32 and            $P3G2
x19 G3 PG32 last2 or          $G3+P3G2

x20 last2 last1 Cout or       $G3+P3G2+P3P2G1+P3P2P1G0
```

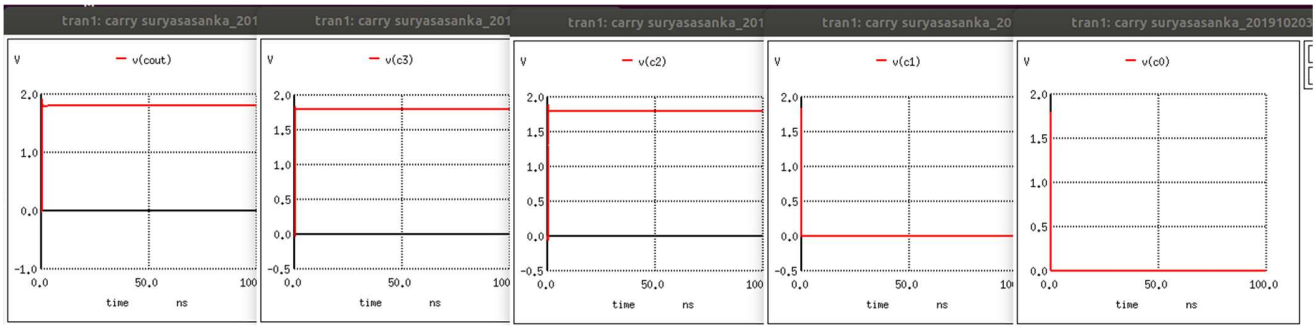
Plots:



B=1010

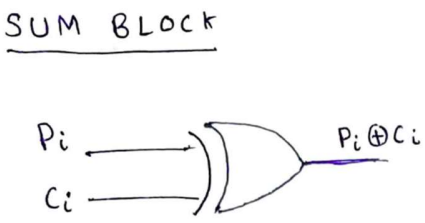


A=1110



Carry=1110 0

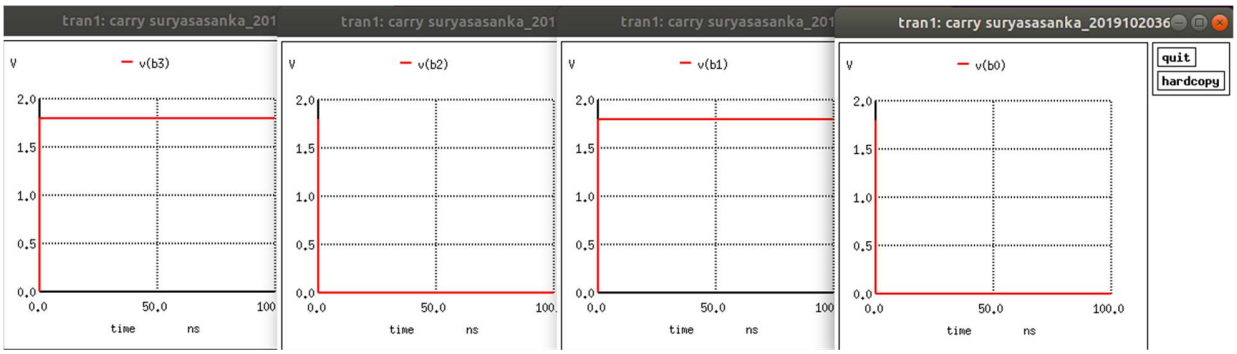
Sum block:



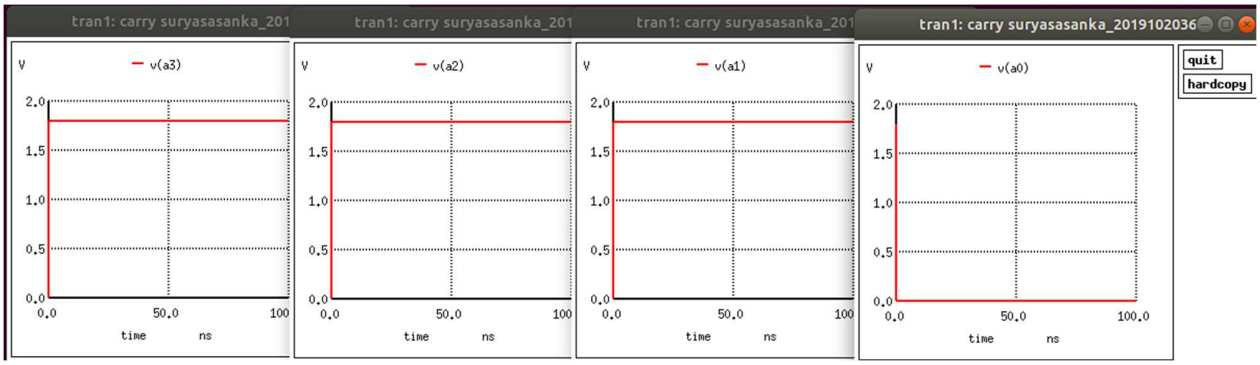
This is the topology of the sum block. Here we just drew for 1 bit. We extend it for all bits.

Ngspice netlist:

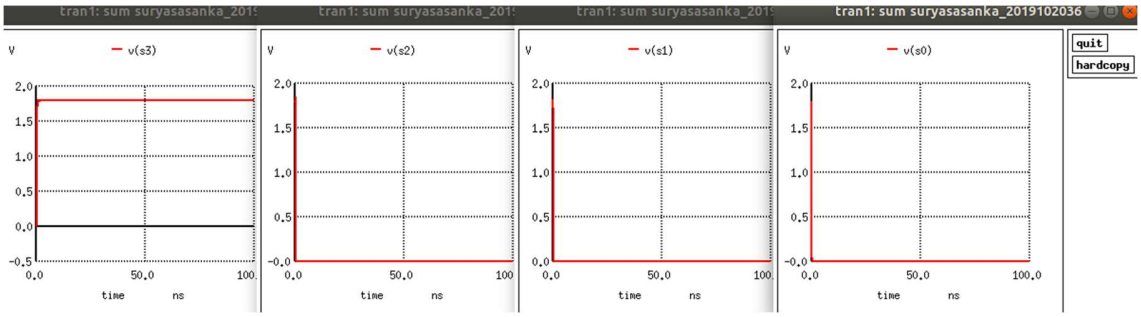
```
*----------------Sum block

x21 P0 C0 S0 xor
x22 P1 C1 S1 xor
x23 P2 C2 S2 xor
x24 P3 C3 S3 xor
```



B=1010

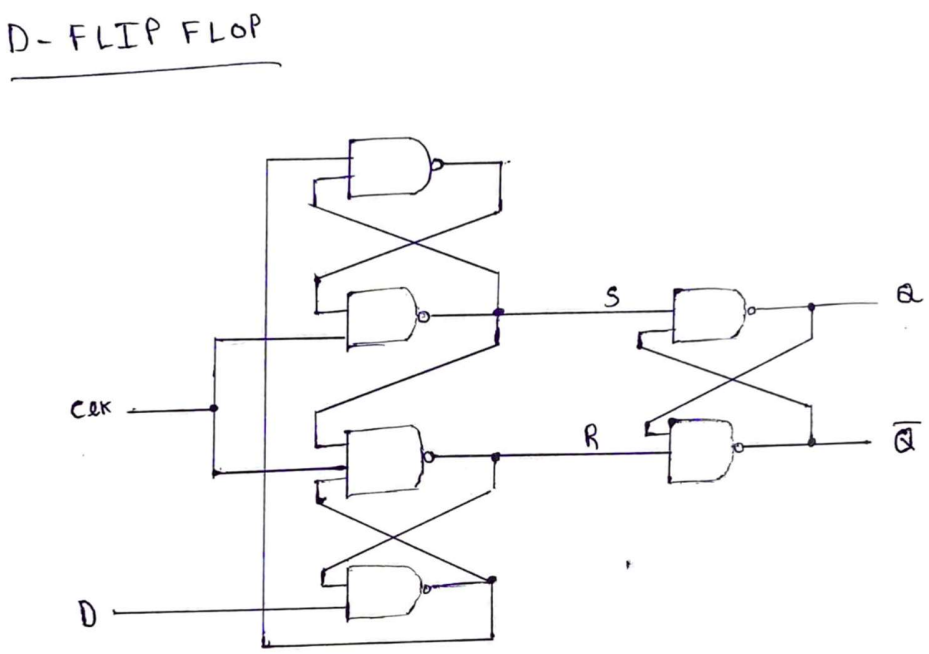


A=1110



Sum=1000

D-flip flop



This is the topology of D-flipflop(+ve edge triggered). The input gets updated at the output for every positive edge of the clock. Here, we additionally use, nand gate with 3 inputs.

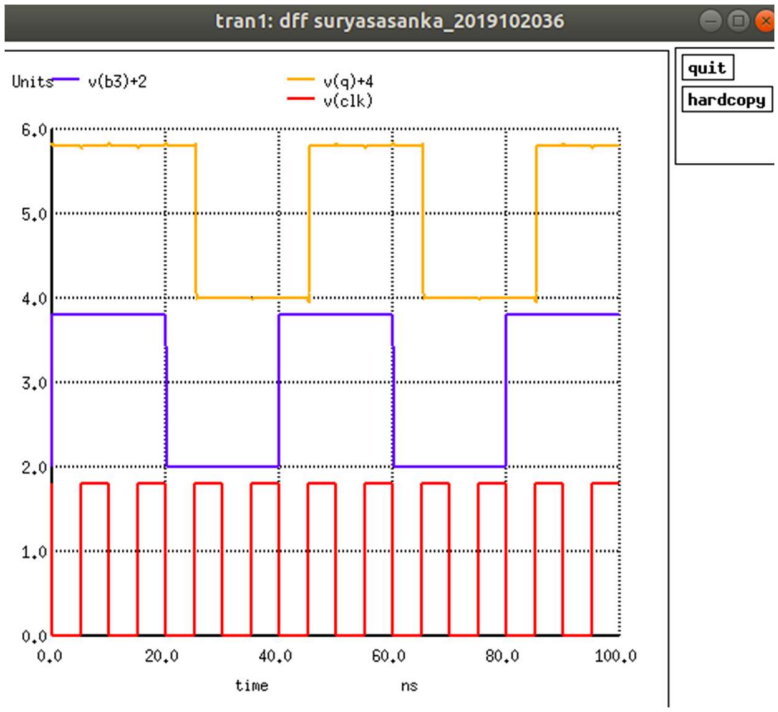
Netlist:

```
.subckt nand3 a b c y
.param width_N={width}
.param width_P={2*width}

M1 y 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 y 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 y 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 y a D1 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 D1 b D2 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 D2 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}

.ends nand3
```

```
.subckt d_ff clk D Q Q_bar

x1 D R R1 nand
x2 R1 clk S R nand3
x3 clk S1 S nand
x4 R1 S S1 nand

x5 S Q_bar Q nand
x6 R Q Q_bar nand


.ends d_ff
```

The drawn circuit diagram is implemented.

Plots:



This plot confirms that our D-flip flop works because whenever there is a positive edge of the clock, we notice that the output takes the value of input and remains there until the next positive edge.

Question-4

Finding setup time, hold time and C-Q delay for the flipflop

Clk-Q delay: It is the time elapsed by the output to reach a stable value(not varying) with respect to the positive clock edge.

Setup time: It is the time before the rising edge of the clock for which the value of the input should be stable(not varying).

Hold time: It is the time after the rising edge of the clock for which the output must be stable(not varying).

The Clk-Q delay can be easily found by using the following netlist snippet.

.measure tran tpd1

+TRIG v(clk) val = 'SUPPLY/2' RISE = 2

+TARG v(Q) val = 'SUPPLY/2' RISE = 1

The value of **Clk-Q delay** is found out to be nearly **1.477 x 10^10 seconds**.

Procedure to find the setup time:

First we must fix the clock pulse. I have taken a clock pulse with time period 10ns.

The second rising edge is at 10ns for the clock. So we change the value of D from 0 to 1 just before the rising edge. We give the transition time to be 9ns. We check the Clk-Q delay.

We slowly increase the transition time by 0.05ns and then check the Clk-Q delay. There is almost no change in the delay.

But when we change the time from 9.85 to 9.86ns, we witness a change >10% in the delay(Sources from internet define that if there is a greater than 10% delay, that is the breaking point). The **setup time** is 10-9.86=**0.14ns**

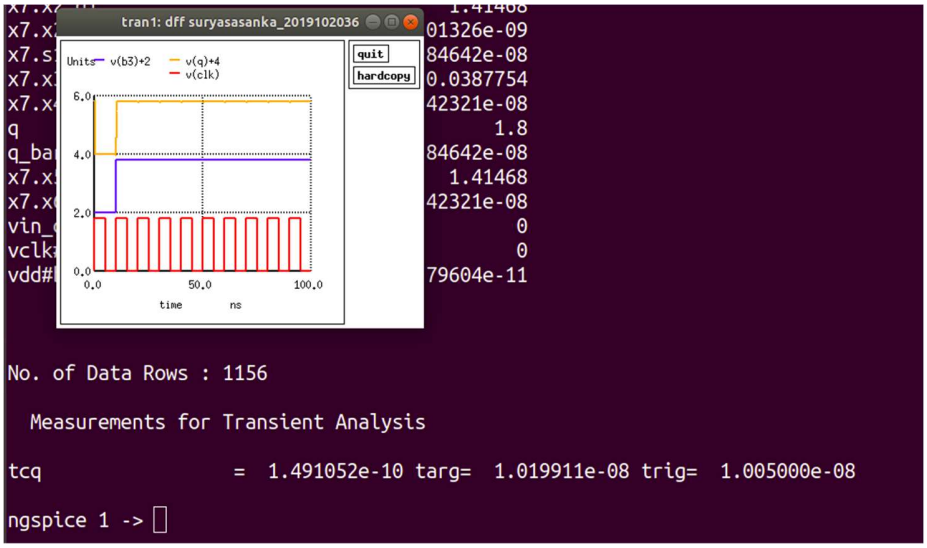
vclk clk 0 pulse 0 1.8 0ns 100ps 100ps 5ns 10ns

Vin_d B3 gnd pwl (0 0v 9.85ns 0v 9.85ns 1.8v 100ns 1.8v)

```
No. of Data Rows : 1157

  Measurements for Transient Analysis

tcq                =  1.477909e-10 targ=  1.019779e-08 trig=  1.005000e-08

ngspice 1 ->
```

vclk clk 0 pulse 0 1.8 0ns 100ps 100ps 5ns 10ns

Vin_d B3 gnd pwl (0 0v 9.86ns 0v 9.86ns 1.8v 100ns 1.8v)



Procedure to find the hold time:

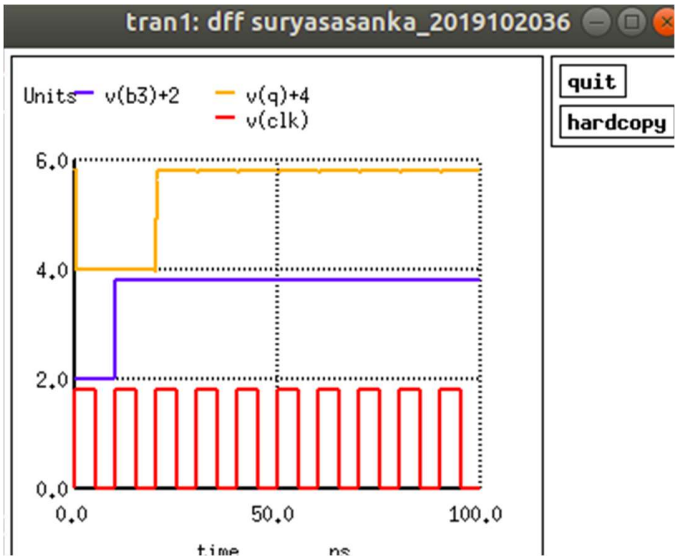
We have considered the same clock used for setup time.

Now, we start at 10.5ns and move towards the 10ns. We notice when we changed D at 10.5 ns, it gets updated at the next rising edge at 15ns.

We move towards left with 0.05 ns precision. We find that even at 10.096ns, the value gets updated at next rising clock edge. However at 10.095ns, the value gets updated at the present rising edge only. So this difference 10.095-10=**0.095ns** is the **hold time**.

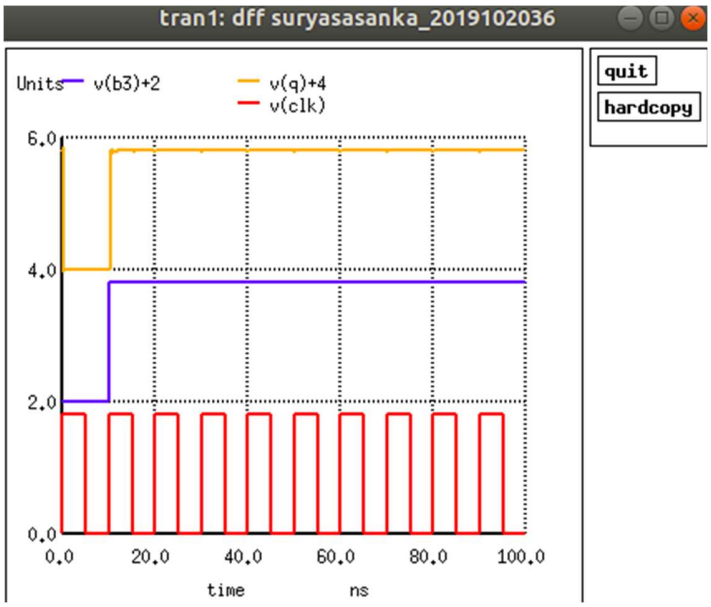
vclk clk 0 pulse 0 1.8 0ns 100ps 100ps 5ns 10ns

Vin_d B3 gnd pwl (0 0v 10.096ns 0v 10.096ns 1.8v 100ns 1.8v)

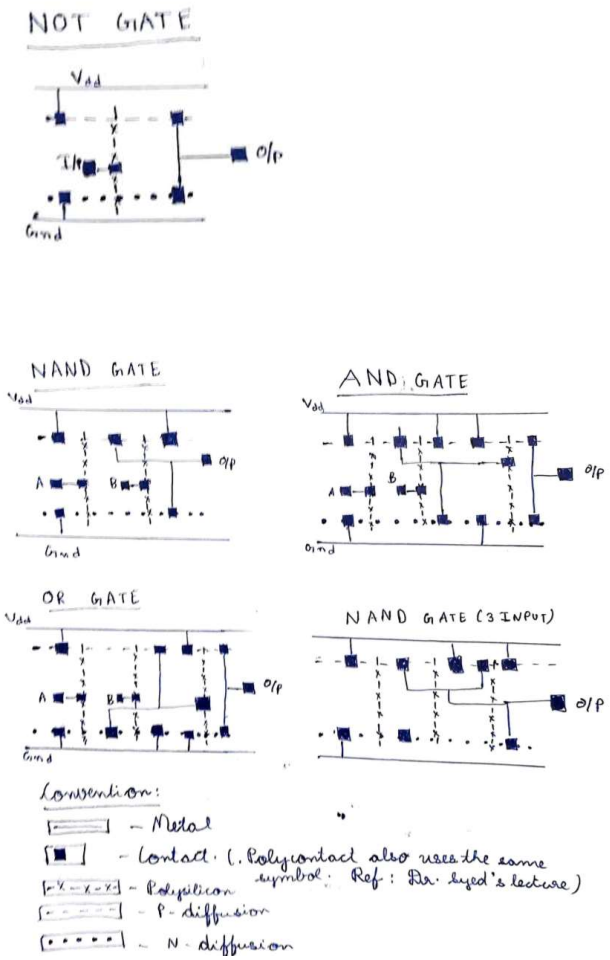


vclk clk 0 pulse 0 1.8 0ns 100ps 100ps 5ns 10ns

Vin_d B3 gnd pwl (0 0v 10.095ns 0v 10.095ns 1.8v 100ns 1.8v)



Question-5



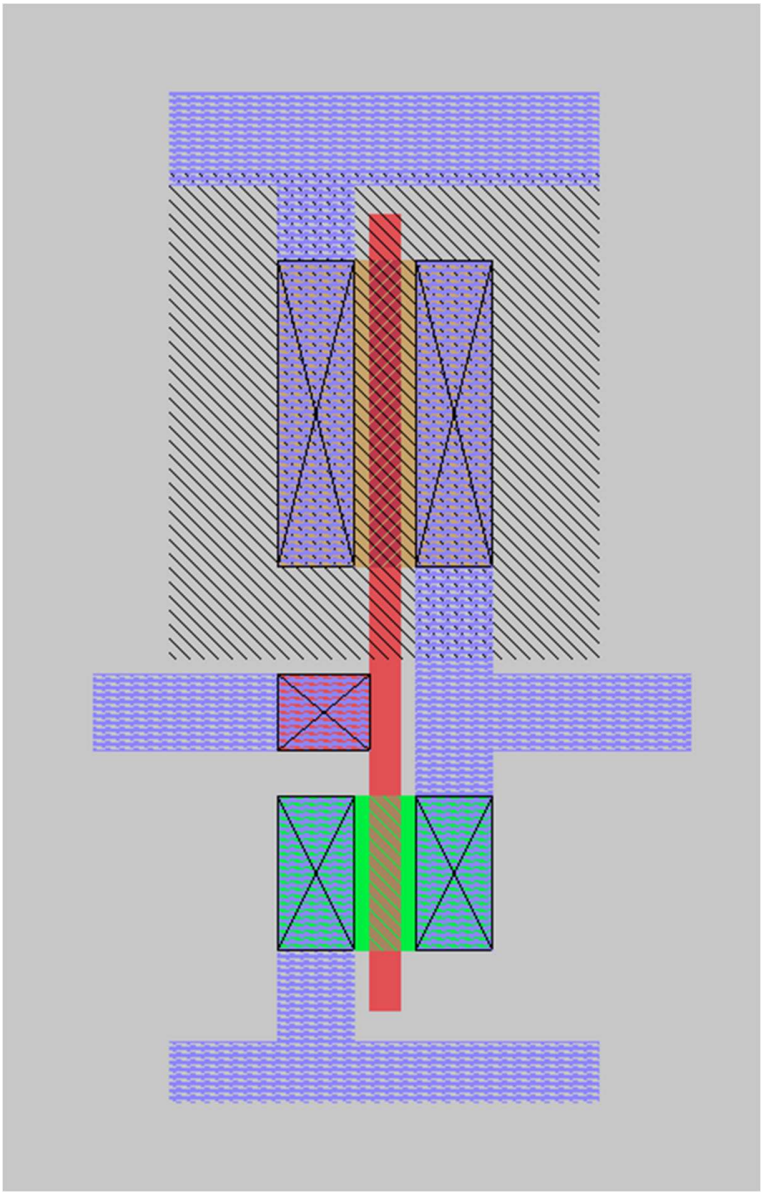
Question-6

We need to layout each block in magic, extract and compare the results.

We can only draw the layout of blocks with the help of gates. So first let us look at the magic layouts of the blocks. Note that for all of the following, width of pmos is 20lambda width of nmos is 10lambda, length of the mosfets is 2lambda.

1) Inverter:

Layout:

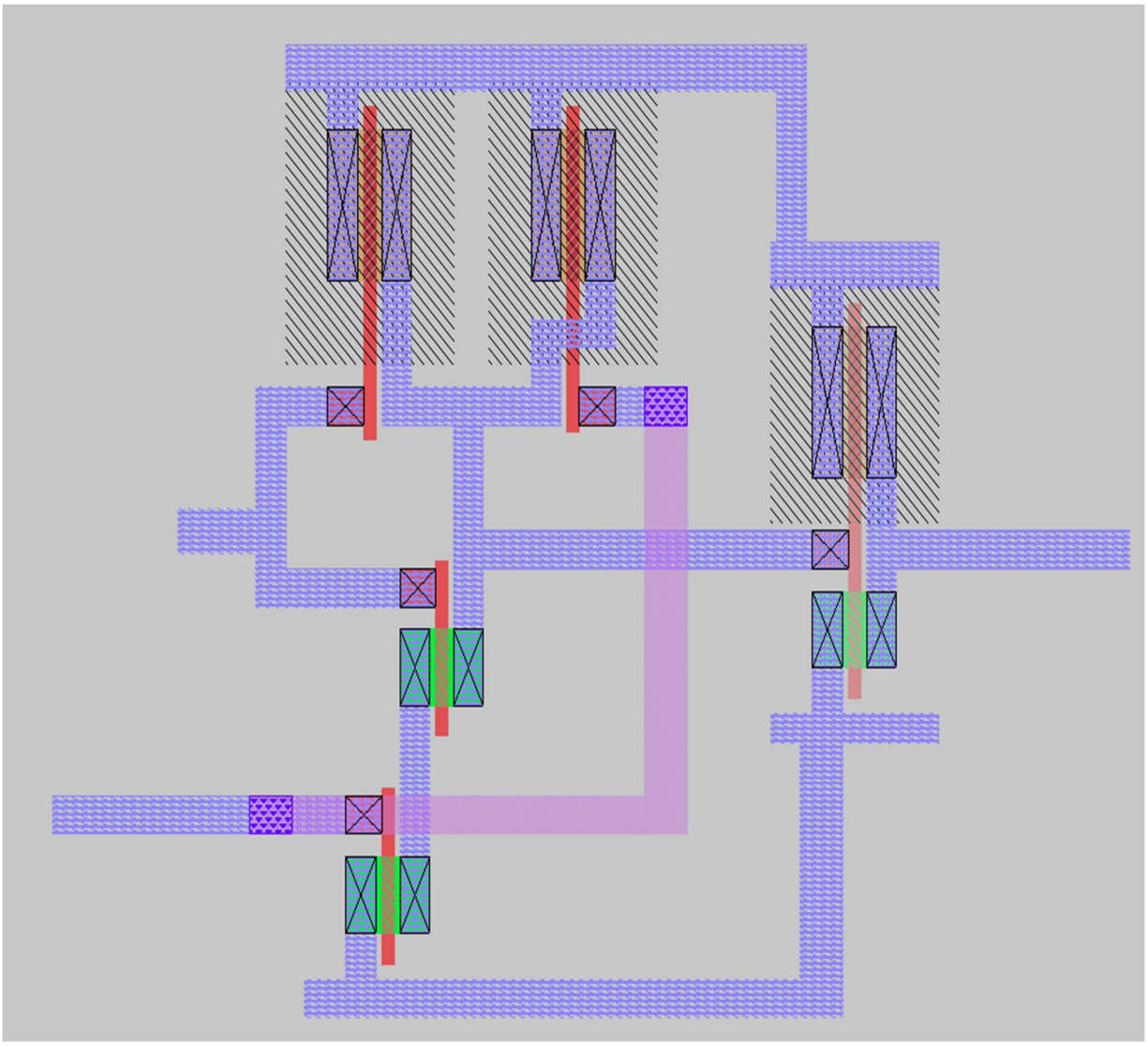


Extracted netlist:

```
Vdd vdd gnd 'SUPPLY'
vin_a input 0 pulse  1.8 0 0ns 100ps 100ps 10ns 20ns

M1 output input vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=120 ps=52
M2 output input gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=60 ps=32

C0 vdd vdd 0.1fF
C1 input vdd 0.0fF
C2 output gnd 0.1fF
C3 vdd output 0.0fF
C4 input output 0.1fF
C5 input gnd 0.1fF
C6 vdd input 0.1fF
C7 vdd output 0.2fF
C8 gnd gnd 0.1fF
C9 output gnd 0.1fF
C10 vdd gnd 0.1fF
C11 input gnd 0.2fF
C12 vdd gnd 0.9fF
```

2) Andgate:

Layout:



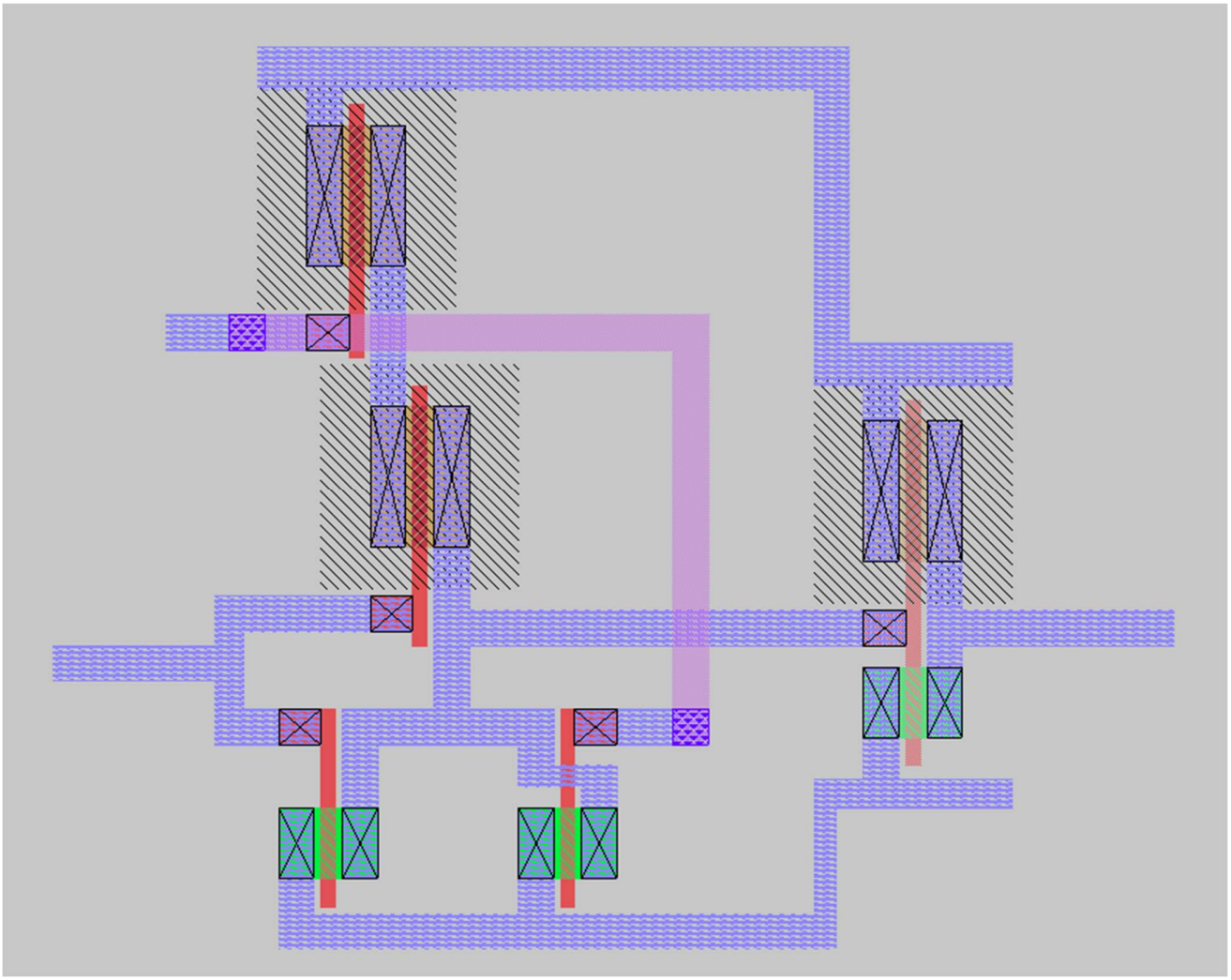
Please that this is not an ideal andgate with minimum capacitances. Changes can be made to see that the height is reduced and nwell to be common.

Extracted netlist:

```
M1000 out a_n61_61# vdd inverter_0/w_n13_n7# CMOSP w=20 l=2
+  ad=120 pd=52 as=360 ps=156
M1001 out a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1002 a_n61_61# clk vdd w_n76_50# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1003 a_n61_61# D vdd w_n42_50# CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1004 a_n61_61# clk a_n58_n25# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1005 a_n58_n25# D gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

C0 inverter_0/w_n13_n7# a_n61_61# 0.1fF
C1 inverter_0/w_n13_n7# vdd 0.1fF
C2 D gnd 0.1fF
C3 a_n58_n25# a_n61_61# 0.1fF
C4 w_n76_50# clk 0.1fF
C5 a_n61_61# vdd 0.6fF
C6 inverter_0/w_n13_n7# out 0.0fF
C7 a_n61_61# out 0.1fF
C8 w_n42_50# D 0.1fF
C9 a_n61_61# gnd 0.1fF
C10 vdd out 0.2fF
C11 a_n58_n25# gnd 0.1fF
C12 w_n76_50# a_n61_61# 0.1fF
C13 out gnd 0.1fF
C14 w_n76_50# vdd 0.1fF
C15 w_n42_50# a_n61_61# 0.1fF
C16 w_n42_50# vdd 0.1fF
C17 clk a_n61_61# 0.1fF
C18 clk a_n58_n25# 0.1fF
C19 D a_n61_61# 0.3fF
C20 D a_n58_n25# 0.2fF
C21 a_n58_n25# gnd 0.1fF
C22 D gnd 1.8fF
C23 clk gnd 0.5fF
C24 w_n42_50# gnd 1.0fF
C25 w_n76_50# gnd 1.0fF
C26 gnd gnd 0.6fF
C27 out gnd 0.2fF
C28 vdd gnd 0.4fF
C29 a_n61_61# gnd 0.6fF
C30 inverter_0/w_n13_n7# gnd 0.9fF
```

3) Or gate:

Layout:



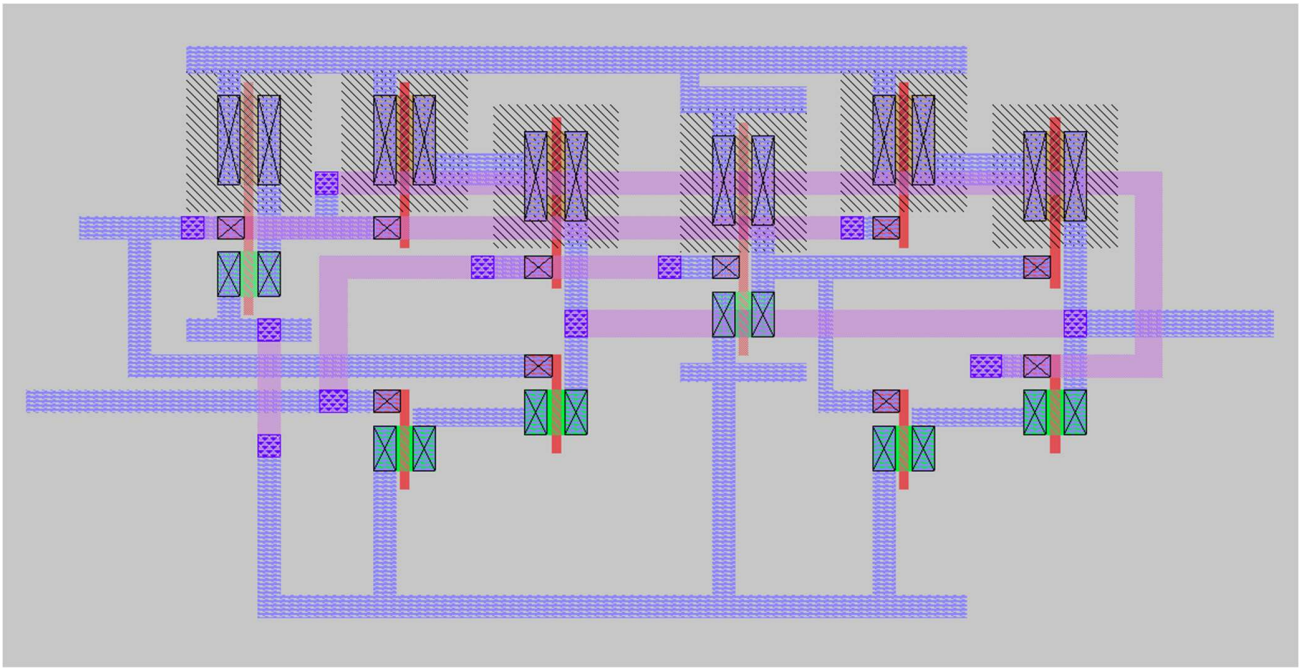
Please that this is not an ideal orgate with minimum capacitances. Changes can be made to see that the height is reduced and nwell to be common.

Extracted netlist:

```
M1 y y_bar vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=240 ps=104
M2 y y_bar gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=180 ps=96
M3 D1 a vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M4 y_bar b D1 vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M5 y_bar b gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M6 y_bar a gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

C0 D1 b 0.0fF
C1 vdd D1 0.0fF
C2 y_bar D1 0.2fF
C3 vdd a 0.0fF
C4 vdd y_bar 0.1fF
C5 vdd D1 0.2fF
C6 y_bar b 0.1fF
C7 vdd vdd 0.1fF
```

4) Xorgate:

Layout:



Even this xor gate is not ideal gate. Changes can be made to see that the capacitances get reduced.

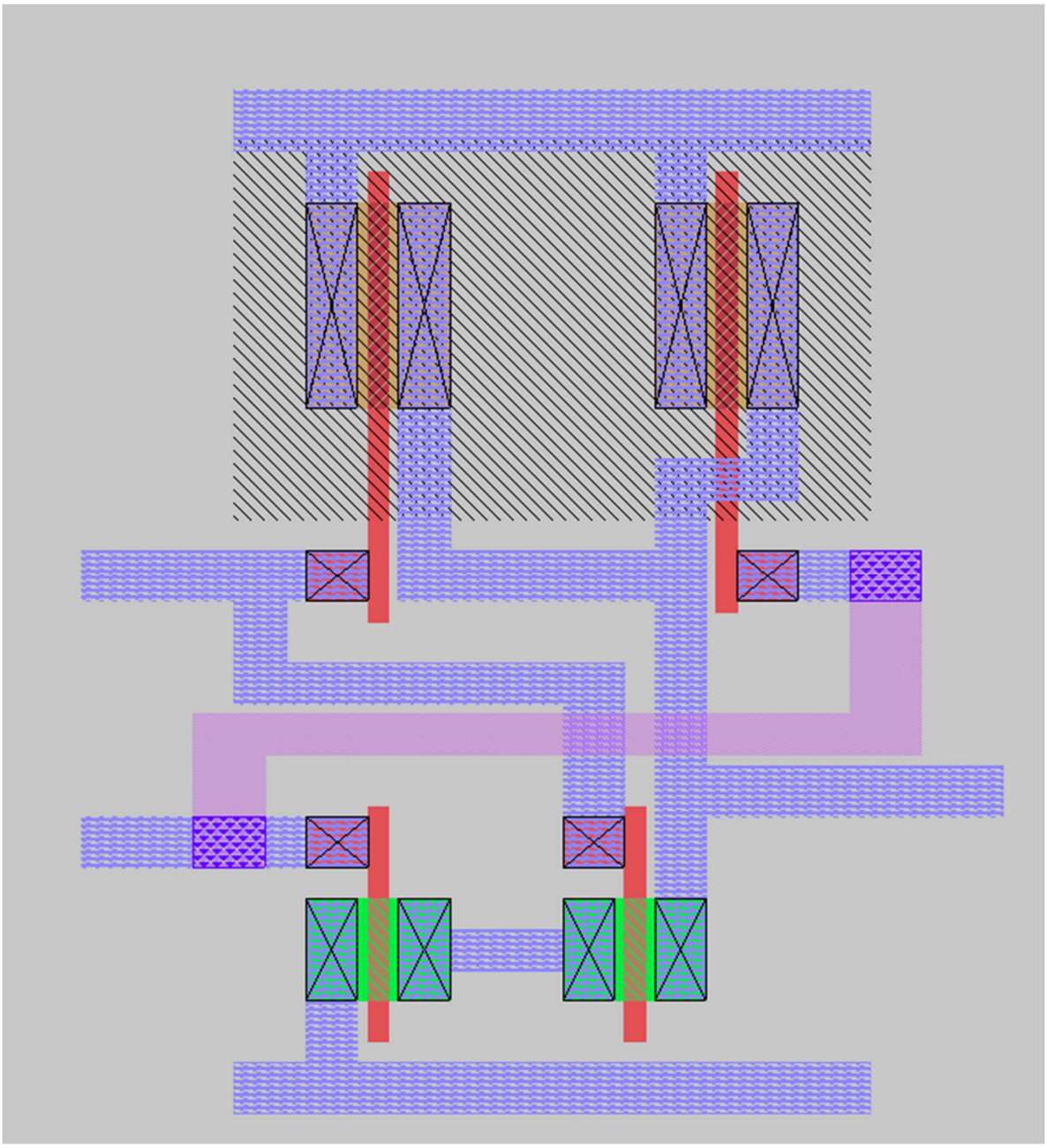
Extracted netlist:

```
M1 B_bar B vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=480 ps=208
M2 B_bar B gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=240 ps=128
M3 A_bar A vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M4 A_bar A gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M5 a_n56_44# A_bar vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M6 Y B a_n56_44# vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M7 a_56_44# A vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M8 Y B_bar a_56_44# vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M9 a_n56_n20# B gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M10 Y A a_n56_n20# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M11 a_56_n20# B_bar gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M12 Y A_bar a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

C0 vdd a_n56_44# 0.1fF
C1 A_bar a_56_44# 0.4fF
C2 A vdd 0.1fF
C3 A_bar vdd 0.2fF
C4 vdd vdd 0.1fF
C5 Y a_56_44# 0.2fF
C6 B a_n56_n20# 0.0fF
C7 A_bar vdd 0.1fF
C8 vdd B 0.1fF
C9 vdd A_bar 0.0fF
C10 vdd Y 0.0fF
C11 A a_n56_n20# 0.2fF
C12 A_bar vdd 0.2fF
C13 B_bar vdd 0.1fF
C14 vdd A 0.0fF
C15 vdd vdd 0.1fF
C16 A_bar vdd 0.1fF
C17 B A 1.2fF
```

Note that there are more capacitances which are omitted in the image.

5) Nandgate(two input):

Layout:



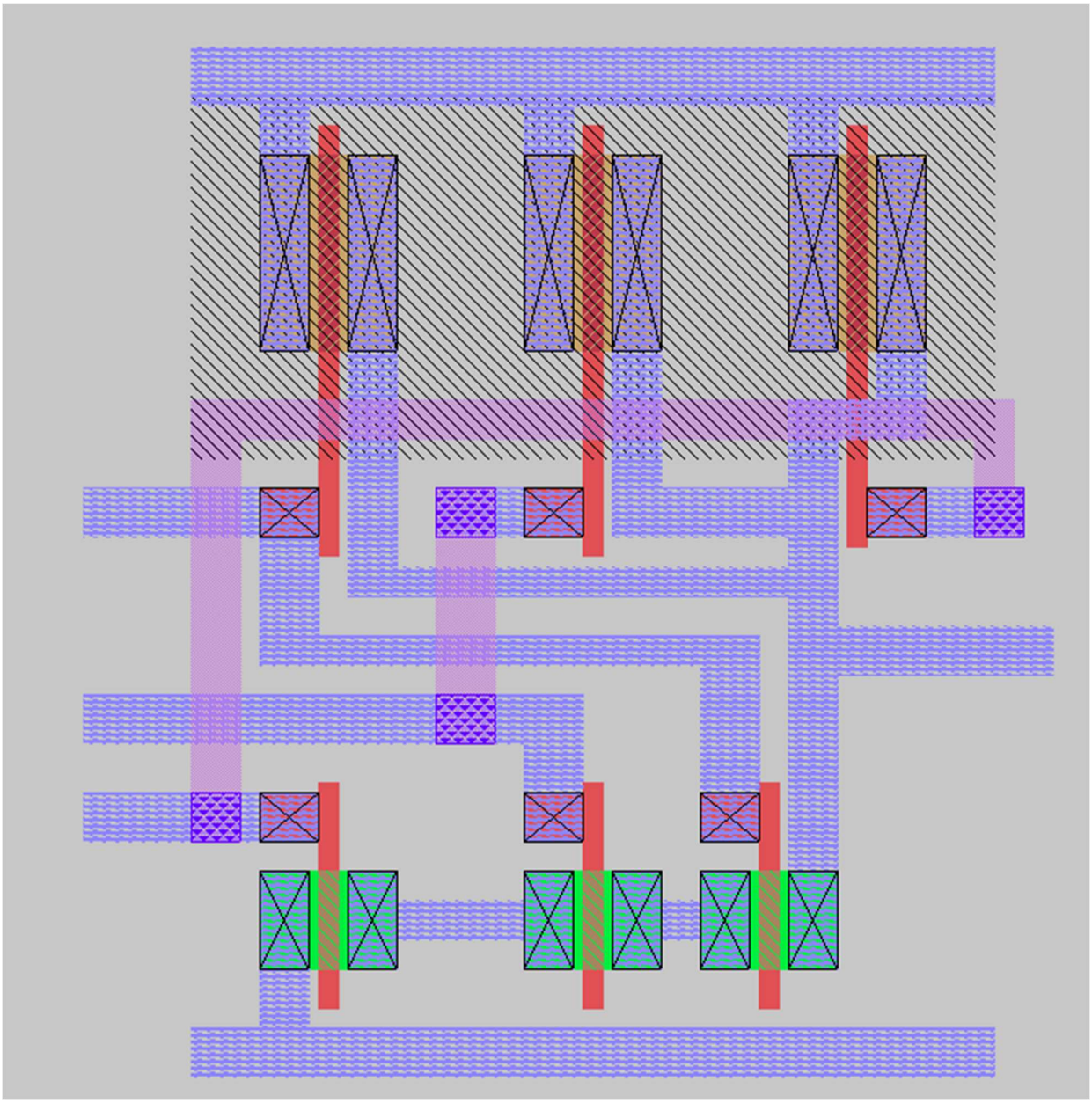
Note that this is a near ideal nandgate drawn in minimum area and resulting in less number of capacitances.

Extracted netlist:

```
M1000 Out B vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=240 ps=104
M1001 Out A vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1002 a_137_45# A gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=60 ps=32
M1003 Out B a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
C0 Out gnd 0.0fF
C1 A Out 0.3fF
C2 Out a_137_45# 0.1fF
C3 A gnd 0.1fF
C4 vdd vdd 0.2fF
C5 gnd a_137_45# 0.2fF
C6 vdd Out 0.2fF
C7 B Out 0.5fF
C8 A vdd 0.1fF
C9 B A 0.1fF
C10 B a_137_45# 0.1fF
C11 B vdd 0.1fF
C12 vdd Out 0.5fF
C13 a_137_45# gnd 0.1fF
C14 gnd gnd 0.3fF
C15 Out gnd 0.3fF
C16 vdd gnd 0.2fF
C17 A gnd 1.3fF
C18 B gnd 0.5fF
C19 vdd gnd 2.3fF
```

6) Nandgate(three input):

Layout:



This is a near ideal 3 input Nand gate with minimum height and less capacitances.

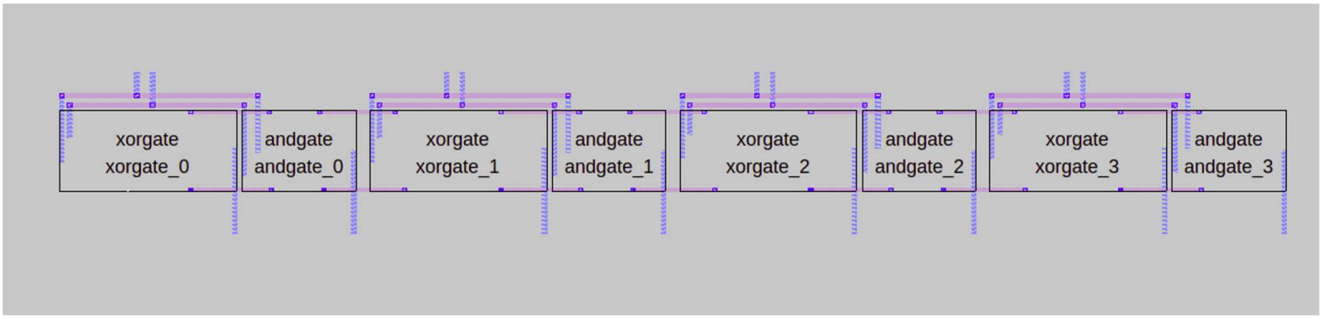
Extracted netlist:

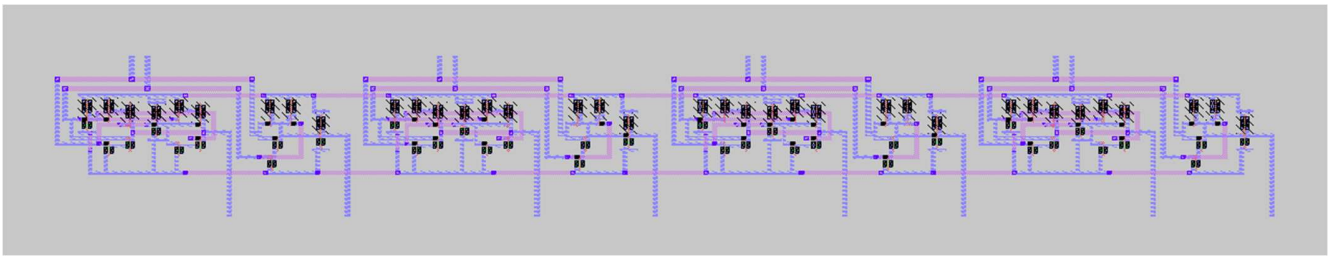
```
M1000 out A vdd vdd CMOSP w=20 l=2
+  ad=360 pd=156 as=360 ps=156
M1001 out B vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1002 out C vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1003 a_79_9# C gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=60 ps=32
M1004 a_106_9# B a_79_9# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1005 out A a_106_9# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0

C0 A out 0.9fF
C1 a_106_9# out 0.1fF
C2 B out 0.4fF
C3 a_106_9# gnd 0.1fF
C4 C out 0.5fF
C5 a_106_9# a_79_9# 0.1fF
C6 B a_79_9# 0.1fF
C7 C gnd 0.1fF
C8 vdd out 0.8fF
C9 out gnd 0.0fF
C10 vdd A 0.1fF
C11 vdd B 0.1fF
C12 a_106_9# A 0.1fF
C13 gnd a_79_9# 0.2fF
C14 A B 0.5fF
C15 vdd C 0.8fF
C16 A C 0.1fF
C17 vdd vdd 0.2fF
C18 vdd out 0.2fF
C19 B C 0.3fF
C20 a_106_9# gnd 0.0fF
C21 a_79_9# gnd 0.1fF
C22 gnd gnd 0.3fF
C23 out gnd 0.5fF
C24 vdd gnd 0.2fF
C25 C gnd 0.8fF
C26 B gnd 0.7fF
C27 A gnd 0.6fF
C28 vdd gnd 3.0fF

.tran 0.1n 100n
```

Now that we have our gates, it helps to build the blocks.

1) P&G block

Layout:





Extracted netlist:

```
M1000 G3 andgate_3/a_n61_61# vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=3360 ps=1456
M1001 G3 andgate_3/a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=1440 ps=768
M1002 andgate_3/a_n61_61# A3 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1003 andgate_3/a_n61_61# B3 vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1004 andgate_3/a_n61_61# A3 andgate_3/a_n58_n25# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1005 andgate_3/a_n58_n25# B3 gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1006 xorgate_3/a_48_n7# A3 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1007 xorgate_3/a_48_n7# A3 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1008 xorgate_3/a_n64_32# B3 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1009 xorgate_3/a_n64_32# B3 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1010 xorgate_3/a_n56_44# xorgate_3/a_n64_32# vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1011 P3 A3 xorgate_3/a_n56_44# vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1012 xorgate_3/a_56_44# B3 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1013 P3 xorgate_3/a_48_n7# xorgate_3/a_56_44# vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1014 xorgate_3/a_n56_n20# A3 gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1015 P3 B3 xorgate_3/a_n56_n20# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1016 xorgate_3/a_56_n20# xorgate_3/a_48_n7# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1017 P3 xorgate_3/a_n64_32# xorgate_3/a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

M1018 G2 andgate_2/a_n61_61# vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1019 G2 andgate_2/a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1020 andgate_2/a_n61_61# A2 vdd vdd CMOSP w=20 l=2
```

```
M1054 G0 andgate_0/a_n61_61# vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1055 G0 andgate_0/a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1056 andgate_0/a_n61_61# A0 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1057 andgate_0/a_n61_61# B0 vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1058 andgate_0/a_n61_61# A0 andgate_0/a_n58_n25# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1059 andgate_0/a_n58_n25# B0 gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1060 xorgate_0/a_48_n7# A0 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1061 xorgate_0/a_48_n7# A0 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1062 xorgate_0/a_n64_32# B0 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1063 xorgate_0/a_n64_32# B0 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1064 xorgate_0/a_n56_44# xorgate_0/a_n64_32# vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1065 P0 A0 xorgate_0/a_n56_44# vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1066 xorgate_0/a_56_44# B0 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1067 P0 xorgate_0/a_48_n7# xorgate_0/a_56_44# vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1068 xorgate_0/a_n56_n20# A0 gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1069 P0 B0 xorgate_0/a_n56_n20# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1070 xorgate_0/a_56_n20# xorgate_0/a_48_n7# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1071 P0 xorgate_0/a_n64_32# xorgate_0/a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

C0 xorgate_3/a_n64_32# xorgate_3/a_n56_44# 0.4fF
C1 vdd vdd 0.1fF
C2 gnd xorgate_0/a_n64_32# 0.1fF
C3 vdd xorgate_2/a_n56_44# 0.2fF
C4 xorgate_3/a_56_n20# gnd 0.1fF
C5 xorgate_2/a_n64_32# xorgate_2/a_56_44# 0.4fF
```

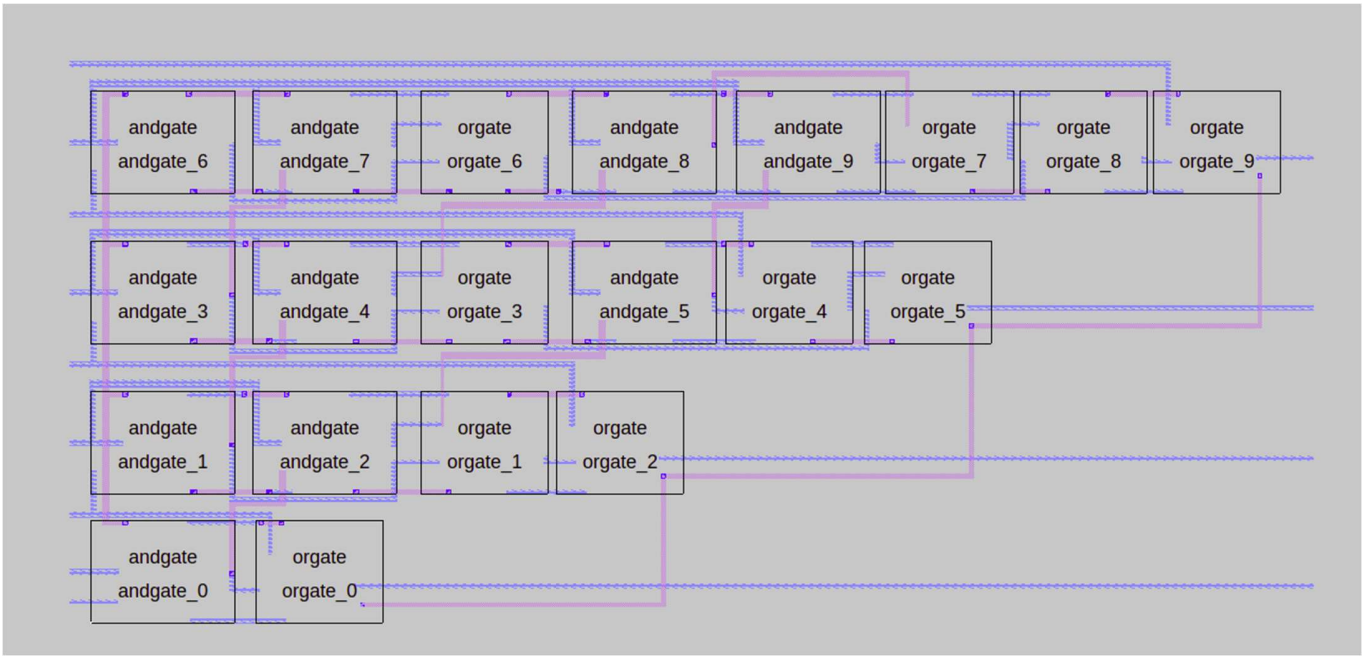
```
C259 vdd gnd 0.9fF
C260 xorgate_3/a_56_n20# gnd 0.1fF
C261 xorgate_3/a_n56_n20# gnd 0.3fF
C262 xorgate_3/a_56_44# gnd 0.0fF
C263 P3 gnd 2.6fF
C264 xorgate_3/a_n56_44# gnd 0.1fF
C265 vdd gnd 0.9fF
C266 vdd gnd 1.0fF
C267 vdd gnd 1.2fF
C268 vdd gnd 0.9fF
C269 xorgate_3/a_n64_32# gnd 1.4fF
C270 vdd gnd 1.0fF
C271 xorgate_3/a_48_n7# gnd 0.8fF
C272 vdd gnd 1.0fF
C273 andgate_3/a_n58_n25# gnd 0.3fF
C274 vdd gnd 1.1fF
C275 vdd gnd 1.1fF
C276 gnd gnd 23.1fF
C277 G3 gnd 0.7fF
C278 vdd gnd 18.1fF
C279 andgate_3/a_n61_61# gnd 1.0fF
C280 vdd gnd 0.9fF
```

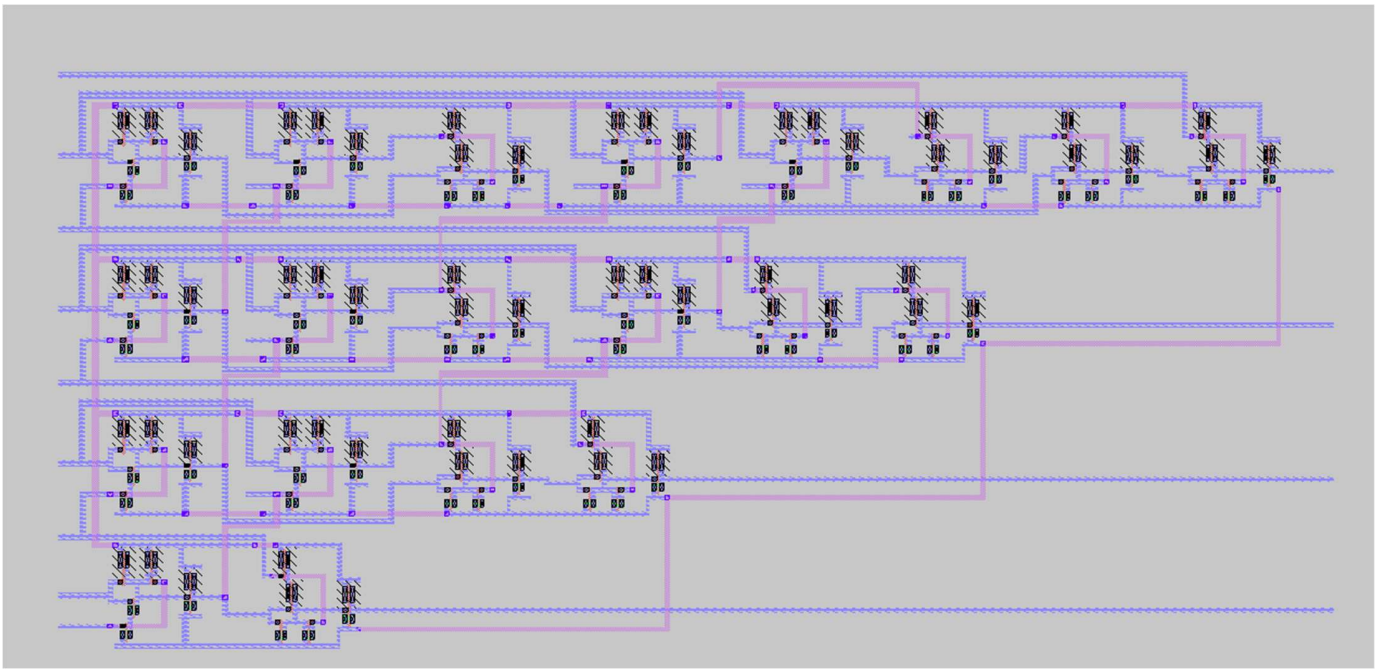
This block is working fine and giving correct results which are the same as the previous ngspice code. Plots are not attached because they are repetitive. The cla(png+carry generate+sum) block will be run and the comparison between the ngspice results and post layout extracted results will be compared in the next sections.

2) Carry generator block

This is the most important and most complicated block in our layout.

Layout:





Extracted netlist:

The extracted netlist contains 120 MOSFETs and 495 capacitors. Only a few are attached.

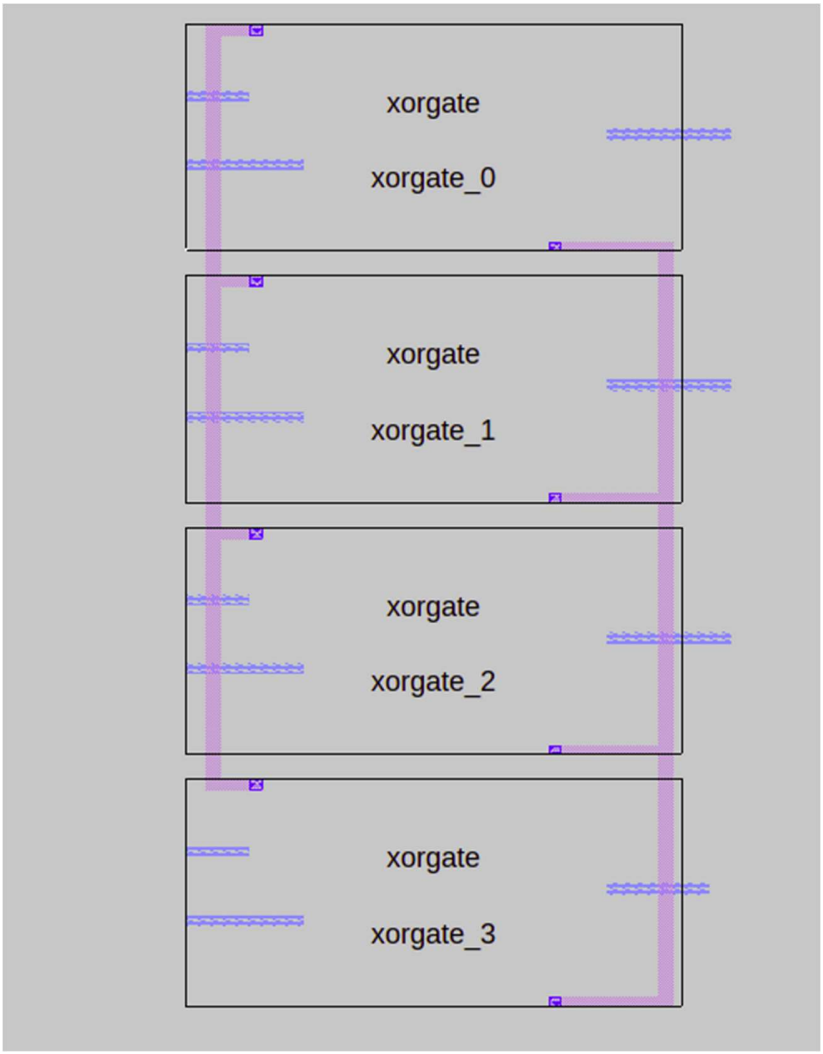
```
+  ad=0 pd=0 as=0 ps=0

M1108 m1_376_596# andgate_7/a_n61_61# vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1109 m1_376_596# andgate_7/a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1110 andgate_7/a_n61_61# P3 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1111 andgate_7/a_n61_61# m1_174_337# vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1112 andgate_7/a_n61_61# P3 andgate_7/a_n58_n25# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1113 andgate_7/a_n58_n25# m1_174_337# gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

M1114 m1_174_525# andgate_6/a_n61_61# vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1115 m1_174_525# andgate_6/a_n61_61# gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1116 andgate_6/a_n61_61# P3 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1117 andgate_6/a_n61_61# G2 vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1118 andgate_6/a_n61_61# P3 andgate_6/a_n58_n25# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=120 ps=64
M1119 andgate_6/a_n58_n25# G2 gnd gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0


C0 vdd orgate_0/a_n63_n10# 0.1fF
C1 m1_174_38# vdd 0.4fF
C2 m1_777_596# andgate_8/a_n61_61# 0.1fF
C3 andgate_9/a_n61_61# vdd 0.1fF
C4 m1_1147_580# orgate_7/a_n63_n10# 0.1fF
C5 andgate_5/a_n61_61# vdd 0.1fF
C6 orgate_5/a_n63_n10# m1_567_341# 0.1fF
C7 orgate_7/a_n63_n10# m1_981_575# 0.1fF
C8 vdd m2_438_434# 0.1fF
C9 gnd m1_376_596# 0.1fF
C10 vdd vdd 0.1fF
C11 andgate_1/a_n58_n25# gnd 0.1fF
C12 andgate_1/a_n61_61# vdd 0.1fF
C13 andgate_0/a_n61_61# gnd 0.1fF
C14 vdd vdd 0.1fF
```

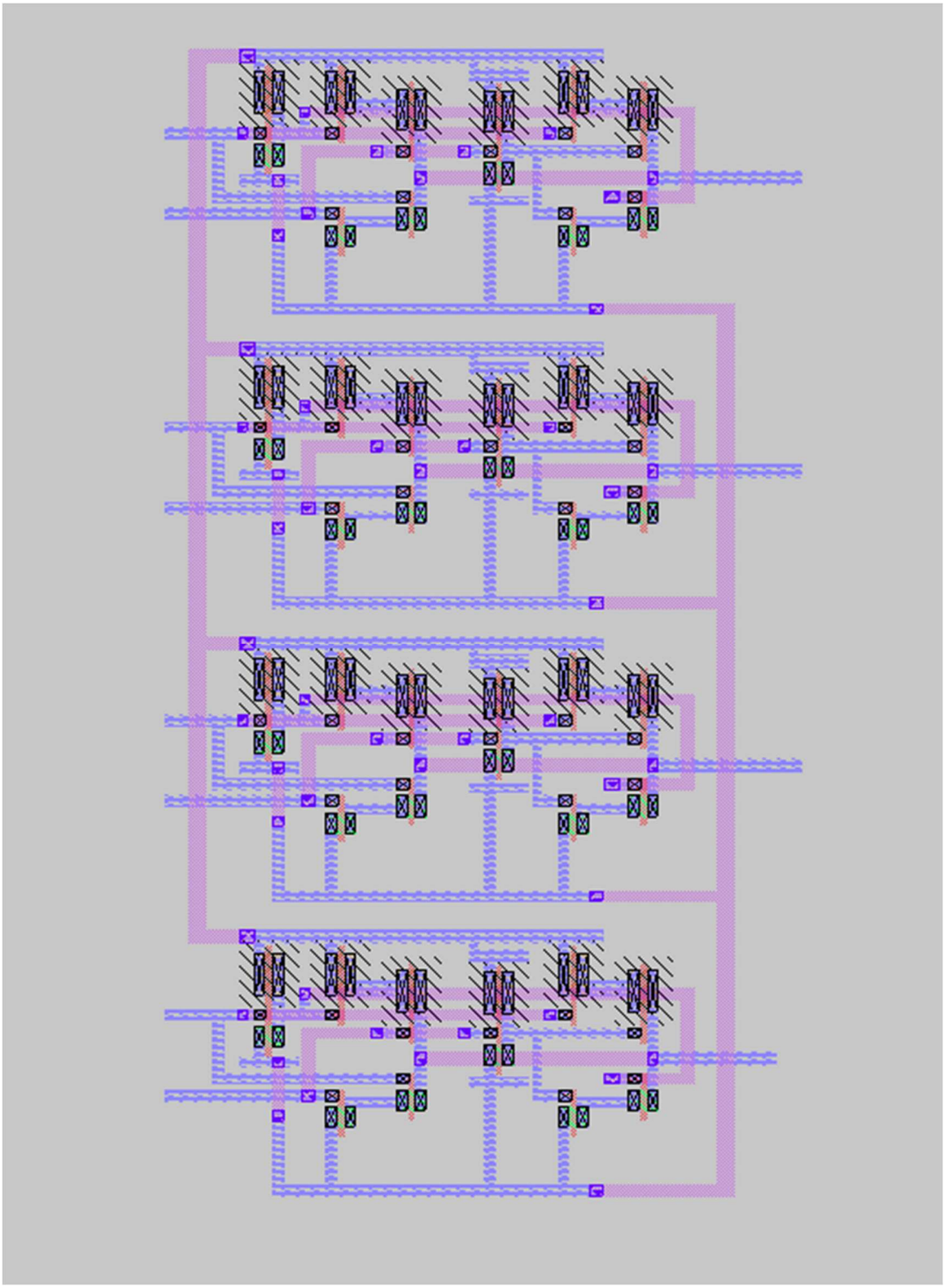
```
C453 vdd gnd 0.9fF
C454 m1_567_341# gnd 1.7fF
C455 orgate_5/a_n59_77# gnd 0.0fF
C456 m1_947_392# gnd 2.1fF
C457 vdd gnd 0.9fF
C458 vdd gnd 0.9fF
C459 orgate_5/a_n63_n10# gnd 0.7fF
C460 vdd gnd 0.9fF
C461 andgate_1/a_n58_n25# gnd 0.3fF
C462 vdd gnd 1.1fF
C463 vdd gnd 1.0fF
C464 andgate_1/a_n61_61# gnd 0.9fF
C465 vdd gnd 0.9fF
C466 andgate_2/a_n58_n25# gnd 0.1fF
C467 m1_174_38# gnd 4.4fF
C468 P1 gnd 2.1fF
C469 vdd gnd 1.0fF
C470 vdd gnd 1.0fF
C471 andgate_2/a_n61_61# gnd 0.6fF
C472 vdd gnd 0.9fF
C473 orgate_1/a_n59_77# gnd 0.0fF
C474 vdd gnd 0.9fF
C475 vdd gnd 0.9fF
C476 orgate_1/a_n63_n10# gnd 0.6fF
C477 vdd gnd 0.9fF
C478 m1_567_199# gnd 0.7fF
C479 orgate_2/a_n59_77# gnd 0.2fF
C480 vdd gnd 0.9fF
C481 vdd gnd 1.0fF
C482 orgate_2/a_n63_n10# gnd 1.0fF
C483 vdd gnd 0.9fF
C484 andgate_0/a_n58_n25# gnd 0.1fF
C485 car0 gnd 1.9fF
C486 P0 gnd 0.7fF
C487 vdd gnd 1.0fF
C488 vdd gnd 1.0fF
C489 andgate_0/a_n61_61# gnd 0.6fF
C490 vdd gnd 0.9fF
C491 orgate_0/a_n59_77# gnd 0.2fF
C492 vdd gnd 0.9fF
C493 vdd gnd 1.0fF
C494 orgate_0/a_n63_n10# gnd 1.0fF
C495 vdd gnd 0.9fF
```

This block is working fine and giving correct results which are the same as the previous ngspice code. Plots are not attached because they are repetitive. The cla(png+carry generate+sum) block will be run and the comparison between the ngspice results and post layout extracted results will be compared in the next sections.

3) Carry generator block

Layout:





Extracted netlist:

We get 48 MOSFETs and 253 capacitors. Only a few are attached.

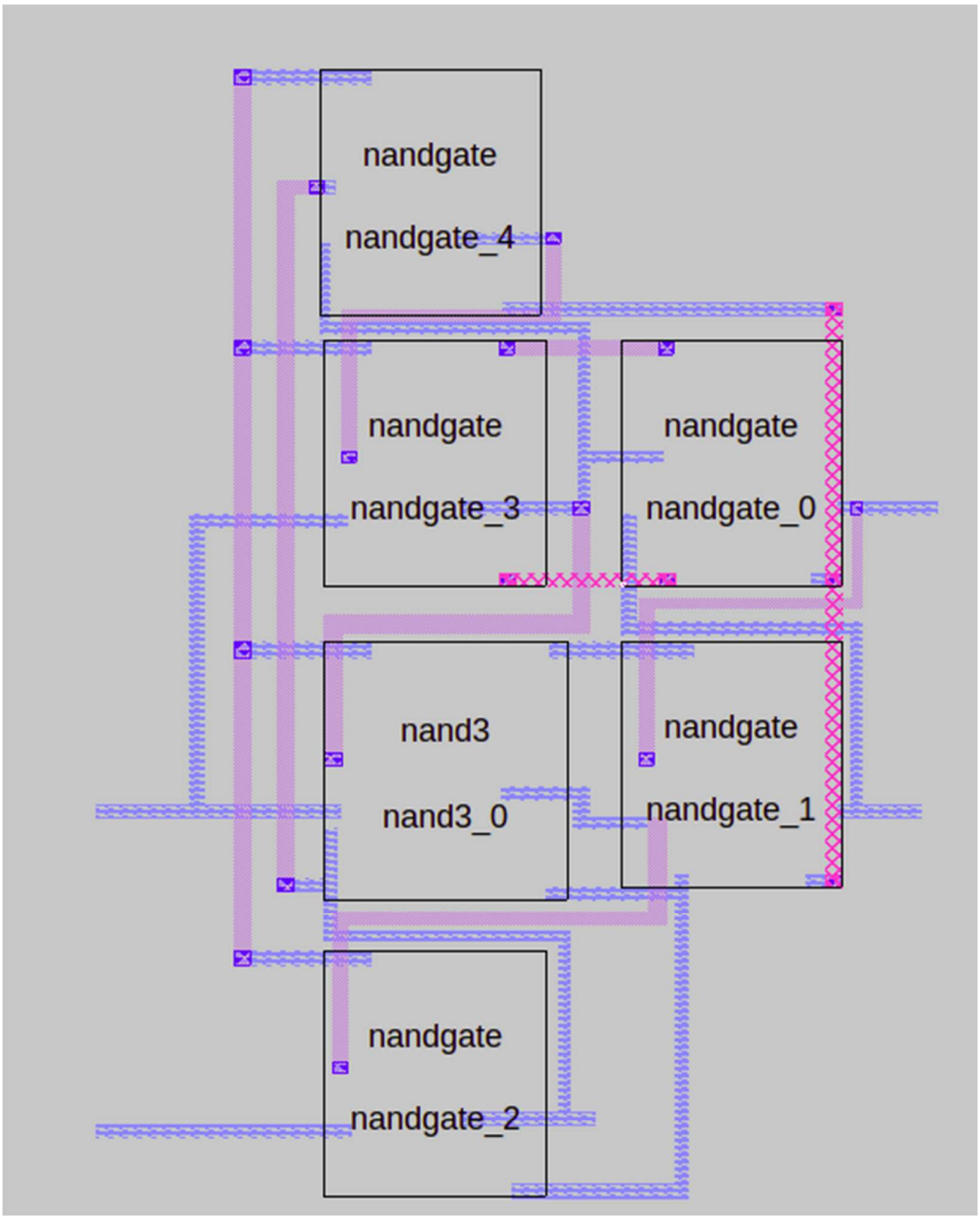
```
+  ad=120 pd=64 as=0 ps=0
M1035 S2 xorgate_1/a_n64_32# xorgate_1/a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

M1036 xorgate_0/a_48_n7# Car3 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1037 xorgate_0/a_48_n7# Car3 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1038 xorgate_0/a_n64_32# P3 vdd vdd CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1039 xorgate_0/a_n64_32# P3 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1040 xorgate_0/a_n56_44# xorgate_0/a_n64_32# vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1041 S3 Car3 xorgate_0/a_n56_44# vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1042 xorgate_0/a_56_44# P3 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1043 S3 xorgate_0/a_48_n7# xorgate_0/a_56_44# vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1044 xorgate_0/a_n56_n20# Car3 gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1045 S3 P3 xorgate_0/a_n56_n20# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1046 xorgate_0/a_56_n20# xorgate_0/a_48_n7# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1047 S3 xorgate_0/a_n64_32# xorgate_0/a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0

C0 P0 vdd 0.1fF
C1 xorgate_3/a_n64_32# vdd 0.2fF
C2 xorgate_2/a_n64_32# xorgate_2/a_56_n20# 0.1fF
C3 vdd xorgate_3/a_n64_32# 0.0fF
C4 xorgate_1/a_n64_32# xorgate_1/a_56_44# 0.4fF
C5 vdd vdd 0.1fF
C6 xorgate_2/a_n64_32# S1 0.6fF
C7 S3 xorgate_0/a_56_44# 0.2fF
C8 xorgate_3/a_56_44# xorgate_3/a_n64_32# 0.4fF
C9 xorgate_0/a_n64_32# xorgate_0/a_56_n20# 0.1fF
C10 vdd xorgate_0/a_56_44# 0.1fF
C11 S2 xorgate_1/a_n56_n20# 0.1fF
C12 vdd xorgate_0/a_n56_44# 0.1fF
C13 P1 vdd 0.2fF
C14 S0 gnd 0.2fF
```

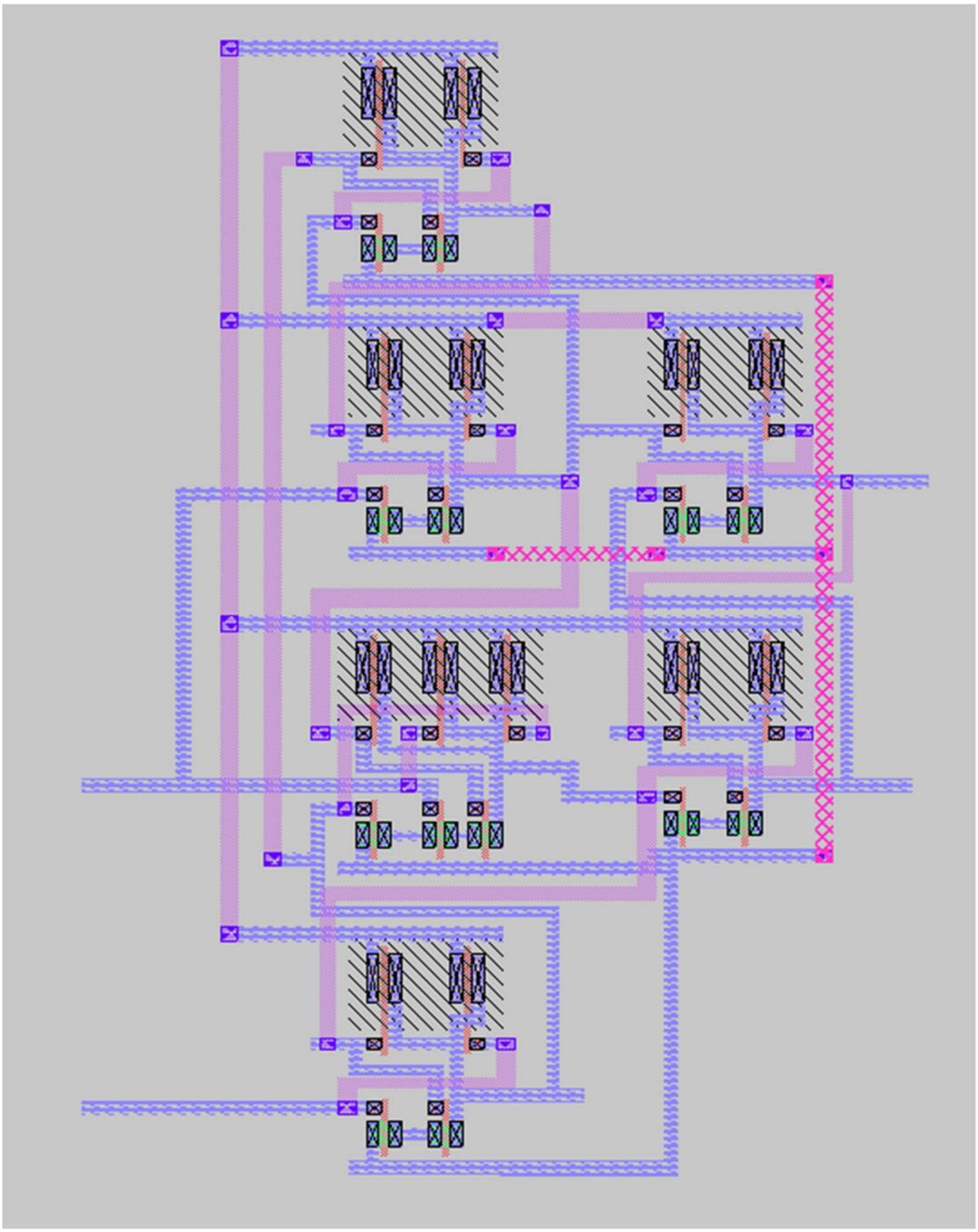
```
C211 vdd gnd 1.0fF
C212 xorgate_1/a_56_n20# gnd 0.1fF
C213 xorgate_1/a_n56_n20# gnd 0.1fF
C214 xorgate_1/a_56_44# gnd 0.0fF
C215 S2 gnd 2.0fF
C216 xorgate_1/a_n56_44# gnd 0.0fF
C217 vdd gnd 0.9fF
C218 vdd gnd 0.9fF
C219 vdd gnd 1.0fF
C220 vdd gnd 0.9fF
C221 xorgate_1/a_n64_32# gnd 1.0fF
C222 P2 gnd 2.8fF
C223 vdd gnd 0.9fF
C224 xorgate_1/a_48_n7# gnd 0.7fF
C225 vdd gnd 1.0fF
C226 xorgate_2/a_56_n20# gnd 0.1fF
C227 xorgate_2/a_n56_n20# gnd 0.1fF
C228 xorgate_2/a_56_44# gnd 0.0fF
C229 S1 gnd 2.0fF
C230 xorgate_2/a_n56_44# gnd 0.0fF
C231 vdd gnd 0.9fF
C232 vdd gnd 0.9fF
C233 vdd gnd 1.0fF
C234 vdd gnd 0.9fF
C235 xorgate_2/a_n64_32# gnd 1.0fF
C236 P1 gnd 2.8fF
C237 vdd gnd 0.9fF
C238 xorgate_2/a_48_n7# gnd 0.7fF
C239 vdd gnd 1.0fF
C240 xorgate_3/a_56_n20# gnd 0.1fF
C241 xorgate_3/a_n56_n20# gnd 0.1fF
C242 xorgate_3/a_56_44# gnd 0.0fF
C243 S0 gnd 2.0fF
C244 xorgate_3/a_n56_44# gnd 0.0fF
C245 vdd gnd 0.9fF
C246 vdd gnd 0.9fF
C247 vdd gnd 1.0fF
C248 vdd gnd 0.9fF
C249 xorgate_3/a_n64_32# gnd 1.0fF
C250 P0 gnd 2.8fF
C251 vdd gnd 0.9fF
C252 xorgate_3/a_48_n7# gnd 0.7fF
C253 vdd gnd 1.0fF
```

This block is working fine and giving correct results which are the same as the previous ngspice code. Plots are not attached because they are repetitive. The cla(png+carry generate+sum) block will be run and the comparison between the ngspice results and post layout extracted results will be compared in the next sections.

4) D-Flipflop

Layout:





Extracted netlist:

```
M1013 R S nand3_0/a_106_9# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0

M1014 Q S vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1015 Q Qbar vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1016 nandgate_0/a_137_45# Qbar gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1017 Q S nandgate_0/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0

M1018 S S1 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1019 S clk vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1020 nandgate_3/a_137_45# clk gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1021 S S1 nandgate_3/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0

M1022 S1 R1 vdd vdd CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1023 S1 S vdd vdd CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1024 nandgate_4/a_137_45# S gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1025 S1 R1 nandgate_4/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0

C0 vdd S1 0.2fF
C1 nandgate_3/a_137_45# gnd 0.2fF
C2 R1 vdd 1.0fF
C3 S Q 0.5fF
C4 R Qbar 0.3fF
C5 vdd Q 0.1fF
C6 R1 S1 0.5fF
C7 S vdd 1.2fF
C8 vdd Q 0.7fF
C9 gnd Qbar 0.3fF
C10 vdd vdd 0.2fF
C11 R1 R 1.2fF
C12 nandgate_2/a_137_45# R 0.1fF
C13 S S1 0.9fF
```

```
C57 vdd S 0.1fF
C58 vdd clk 0.1fF
C59 S Qbar 0.1fF
C60 gnd nand3_0/a_106_9# 0.1fF
C61 Qbar Q 0.9fF
C62 vdd vdd 0.2fF
C63 R1 nandgate_2/a_137_45# 0.1fF
C64 vdd Qbar 0.2fF
C65 vdd vdd 0.2fF
C66 nandgate_3/a_137_45# S1 0.1fF
C67 R1 S 0.2fF
C68 vdd Qbar 0.9fF
C69 nand3_0/a_79_9# clk 0.1fF
C70 gnd nandgate_1/a_137_45# 0.2fF
C71 nand3_0/a_79_9# nand3_0/a_106_9# 0.1fF
C72 R vdd 0.2fF
C73 nandgate_4/a_137_45# gnd 0.1fF
C74 gnd gnd 3.8fF
C75 S1 gnd 2.7fF
C76 vdd gnd 7.0fF
C77 S gnd 5.5fF
C78 R1 gnd 6.1fF
C79 vdd gnd 2.3fF
C80 nandgate_3/a_137_45# gnd 0.1fF
C81 clk gnd 3.0fF
C82 vdd gnd 2.3fF
C83 nandgate_0/a_137_45# gnd 0.1fF
C84 Q gnd 3.1fF
C85 Qbar gnd 2.4fF
C86 vdd gnd 2.3fF
C87 nand3_0/a_106_9# gnd 0.0fF
C88 nand3_0/a_79_9# gnd 0.1fF
C89 vdd gnd 3.0fF
C90 nandgate_1/a_137_45# gnd 0.1fF
C91 R gnd 3.8fF
C92 vdd gnd 2.3fF
C93 nandgate_2/a_137_45# gnd 0.1fF
C94 vdd gnd 2.4fF

.tran 0.1n 100n
.control
set hcopypscolor = 1
set color0=white
set color1=black
```

This flipflop is working as expected. Now, we want to compare prelayout and post layout setup time, hold time and Clk-Q delay.

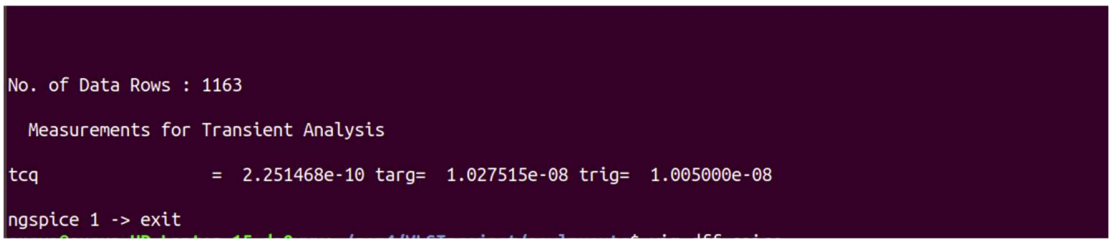
Calculations of times for post layout values:

Clk-Q delay:2.22 x 10^-10

```
No. of Data Rows : 1167

  Measurements for Transient Analysis

tcq                =  2.208765e-10 targ=  1.027088e-08 trig=  1.005000e-08
```

Setup time: At 9.84 and before, we get 2.22 x10^-10 and at 9.85 we get 2.25 x10^-10.

So the **setup time** is 10-9.84=**0.16ns**

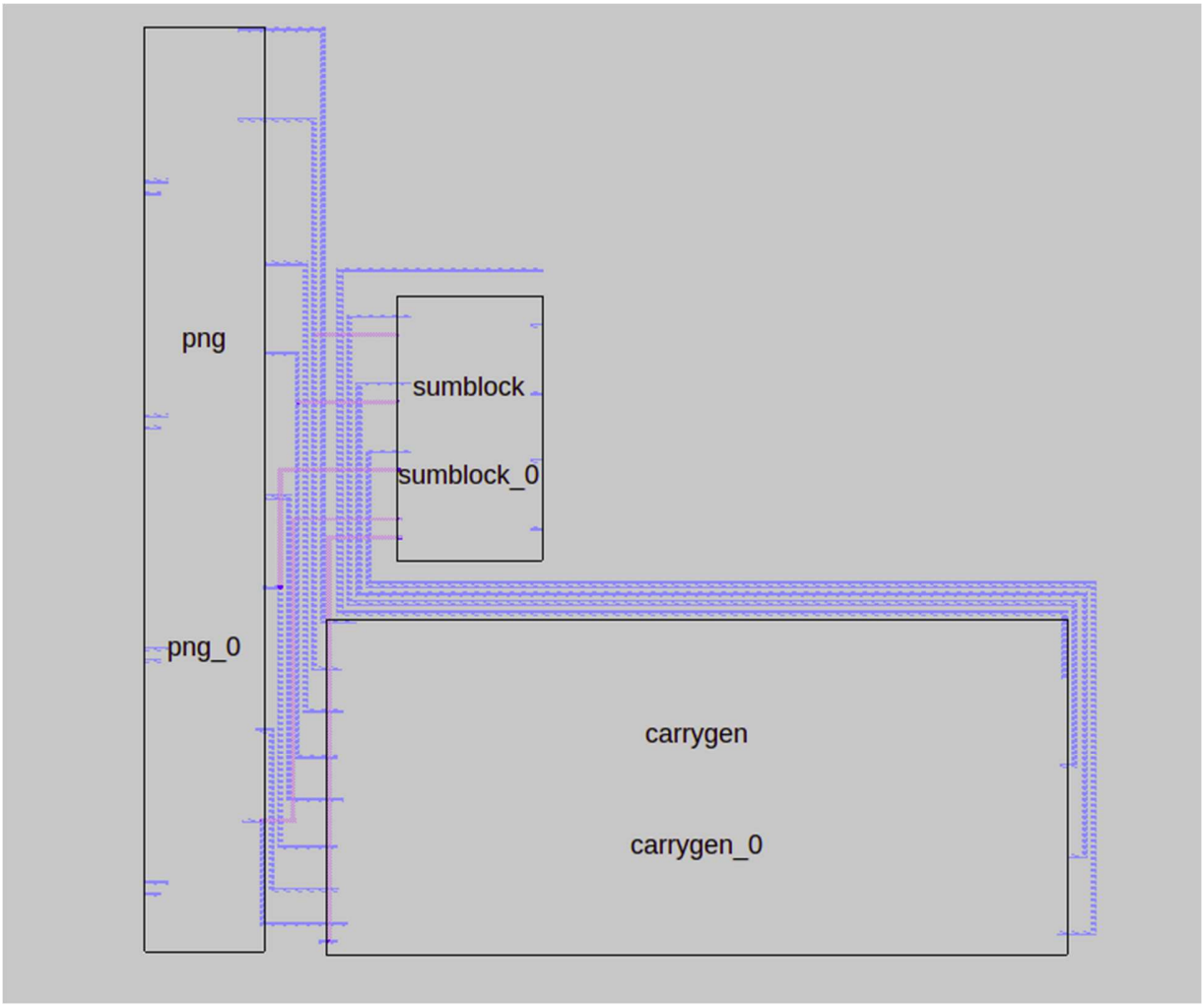


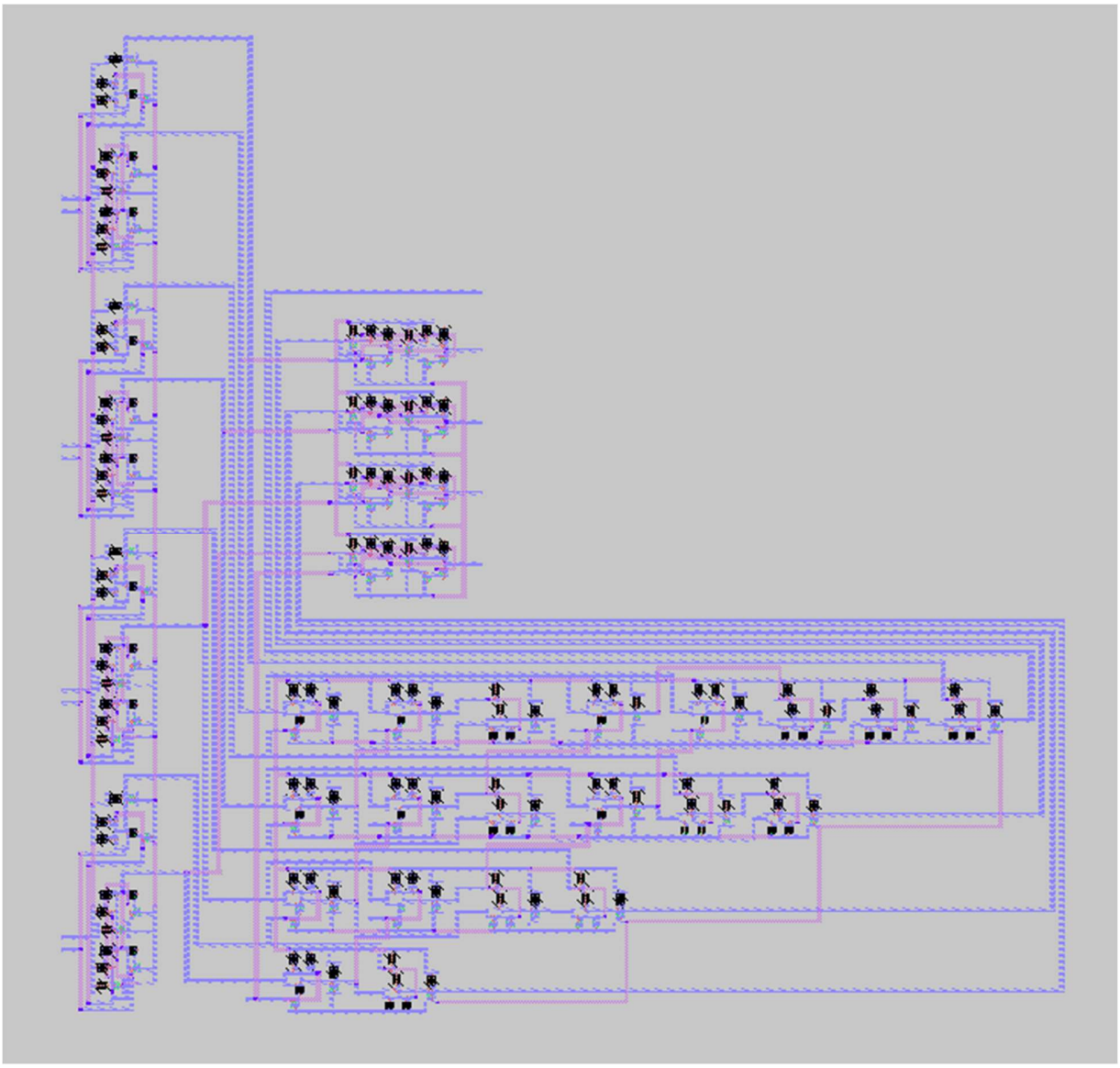
Hold time: At 10.05ns the value gets updated at the same clock edge. At 10.06ns the value gets updated at the next clock edge. So the value of **Hold time** is 10.05-10ns=**0.05ns.**

Since I have not plotted the outputs from post layout simulation and compared to each block of prelayout value, to prove that the circuit is working, I have combined all these blocks and written the CLA. If the values match for CLA(adder without flipflops) when compared with prelayout simulation match, then it logically follows that each of my blocks is correct.

5) CLA (PG+ Carry gen+ Sum)

Layout:





Extracted layout:

We get 240 MOSFETs and 1067 capacitances

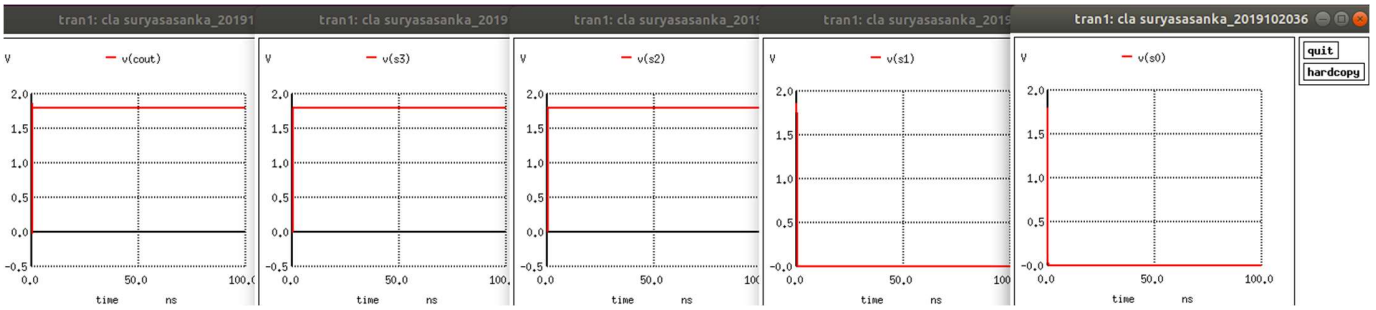
```
+  ad=0 pd=0 as=0 ps=0
M1228 png_0/xorgate_0/a_48_n7# A0 vdd png_0/xorgate_0/inverter_0/w_n13_n7# CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1229 png_0/xorgate_0/a_48_n7# A0 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1230 png_0/xorgate_0/a_n64_32# B0 vdd png_0/xorgate_0/inverter_1/w_n13_n7# CMOSP w=20 l=2
+  ad=120 pd=52 as=0 ps=0
M1231 png_0/xorgate_0/a_n64_32# B0 gnd gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1232 png_0/xorgate_0/a_n56_44# png_0/xorgate_0/a_n64_32# vdd png_0/xorgate_0/w_n71_38# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1233 m1_243_273# A0 png_0/xorgate_0/a_n56_44# png_0/xorgate_0/w_n37_30# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1234 png_0/xorgate_0/a_56_44# B0 vdd png_0/xorgate_0/w_41_38# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1235 m1_243_273# png_0/xorgate_0/a_48_n7# png_0/xorgate_0/a_56_44# png_0/xorgate_0/w_75_30# CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1236 png_0/xorgate_0/a_n56_n20# A0 gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1237 m1_243_273# B0 png_0/xorgate_0/a_n56_n20# gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1238 png_0/xorgate_0/a_56_n20# png_0/xorgate_0/a_48_n7# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1239 m1_243_273# png_0/xorgate_0/a_n64_32# png_0/xorgate_0/a_56_n20# gnd CMOSN w=10 l=2
+  ad=0 pd=0 as=0 ps=0
C0 sumblock_0/xorgate_0/w_n37_30# m1_198_1746# 0.1fF
C1 vdd png_0/xorgate_2/inverter_1/w_n13_n7# 0.1fF
C2 png_0/xorgate_1/a_48_n7# png_0/xorgate_1/w_75_30# 0.1fF
C3 gnd m1_198_1746# 0.1fF
C4 sumblock_0/xorgate_1/a_56_n20# sumblock_0/xorgate_1/a_n64_32# 0.1fF
C5 carrygen_0/andgate_5/inverter_0/w_n13_n7# vdd 0.1fF
C6 sumblock_0/xorgate_0/a_n64_32# sumblock_0/xorgate_0/inverter_1/w_n13_n7# 0.0fF
C7 carrygen_0/m1_1315_575# carrygen_0/orgate_8/inverter_0/w_n13_n7# 0.0fF
C8 carrygen_0/orgate_1/w_n65_31# carrygen_0/m1_174_152# 0.1fF
C9 vdd carrygen_0/m1_981_575# 0.2fF
C10 gnd carrygen_0/andgate_9/a_n61_61# 0.1fF
C11 carrygen_0/andgate_2/a_n61_61# carrygen_0/andgate_2/w_n76_50# 0.1fF
C12 sumblock_0/xorgate_0/a_n64_32# gnd 0.1fF
C13 sumblock_0/xorgate_3/a_n56_44# m1_243_273# 0.1fF
C14 png_0/xorgate_2/inverter_0/w_n13_n7# png_0/xorgate_2/a_48_n7# 0.0fF
C15 carrygen_0/orgate_9/w_n65_31# carrygen_0/m1_1315_575# 0.1fF
C16 carrygen_0/orgate_4/a_n63_n10# carrygen_0/orgate_4/inverter_0/w_n13_n7# 0.1fF
C17 carrygen_0/orgate_0/inverter_0/w_n13_n7# carrygen_0/orgate_0/a_n63_n10# 0.1fF
C18 vdd sumblock_0/xorgate_2/a_n64_32# 0.5fF
```

```
C1025 carrygen_0/orgate_4/w_n65_31# gnd 0.9fF
C1026 carrygen_0/orgate_4/w_n74_71# gnd 0.9fF
C1027 carrygen_0/orgate_4/a_n63_n10# gnd 0.6fF
C1028 carrygen_0/orgate_4/inverter_0/w_n13_n7# gnd 0.9fF
C1029 carrygen_0/m1_567_341# gnd 1.7fF
C1030 carrygen_0/orgate_5/a_n59_77# gnd 0.0fF
C1031 carrygen_0/m1_947_392# gnd 2.1fF
C1032 carrygen_0/orgate_5/w_n65_31# gnd 0.9fF
C1033 carrygen_0/orgate_5/w_n74_71# gnd 0.9fF
C1034 carrygen_0/orgate_5/a_n63_n10# gnd 0.7fF
C1035 carrygen_0/orgate_5/inverter_0/w_n13_n7# gnd 0.9fF
C1036 carrygen_0/andgate_1/a_n58_n25# gnd 0.1fF
C1037 carrygen_0/andgate_1/w_n42_50# gnd 1.0fF
C1038 carrygen_0/andgate_1/w_n76_50# gnd 1.0fF
C1039 carrygen_0/andgate_1/a_n61_61# gnd 0.6fF
C1040 carrygen_0/andgate_1/inverter_0/w_n13_n7# gnd 0.9fF
C1041 carrygen_0/andgate_2/a_n58_n25# gnd 0.1fF
C1042 carrygen_0/m1_174_38# gnd 4.2fF
C1043 carrygen_0/andgate_2/w_n42_50# gnd 1.0fF
C1044 carrygen_0/andgate_2/w_n76_50# gnd 1.0fF
C1045 carrygen_0/andgate_2/a_n61_61# gnd 0.6fF
C1046 carrygen_0/andgate_2/inverter_0/w_n13_n7# gnd 0.9fF
C1047 carrygen_0/orgate_1/a_n59_77# gnd 0.0fF
C1048 carrygen_0/orgate_1/w_n65_31# gnd 0.9fF
C1049 carrygen_0/orgate_1/w_n74_71# gnd 0.9fF
C1050 carrygen_0/orgate_1/a_n63_n10# gnd 0.6fF
C1051 carrygen_0/orgate_1/inverter_0/w_n13_n7# gnd 0.9fF
C1052 carrygen_0/m1_567_199# gnd 0.7fF
C1053 carrygen_0/orgate_2/a_n59_77# gnd 0.0fF
C1054 carrygen_0/orgate_2/w_n65_31# gnd 0.9fF
C1055 carrygen_0/orgate_2/w_n74_71# gnd 0.9fF
C1056 carrygen_0/orgate_2/a_n63_n10# gnd 0.7fF
C1057 carrygen_0/orgate_2/inverter_0/w_n13_n7# gnd 0.9fF
C1058 carrygen_0/andgate_0/a_n58_n25# gnd 0.3fF
C1059 carrygen_0/andgate_0/w_n42_50# gnd 1.1fF
C1060 carrygen_0/andgate_0/w_n76_50# gnd 1.0fF
C1061 carrygen_0/andgate_0/a_n61_61# gnd 0.9fF
C1062 carrygen_0/andgate_0/inverter_0/w_n13_n7# gnd 0.9fF
C1063 carrygen_0/orgate_0/a_n59_77# gnd 0.0fF
C1064 carrygen_0/orgate_0/w_n65_31# gnd 0.9fF
C1065 carrygen_0/orgate_0/w_n74_71# gnd 0.9fF
C1066 carrygen_0/orgate_0/a_n63_n10# gnd 0.7fF
C1067 carrygen_0/orgate_0/inverter_0/w_n13_n7# gnd 0.9fF
.tran 0.1n 100n
```

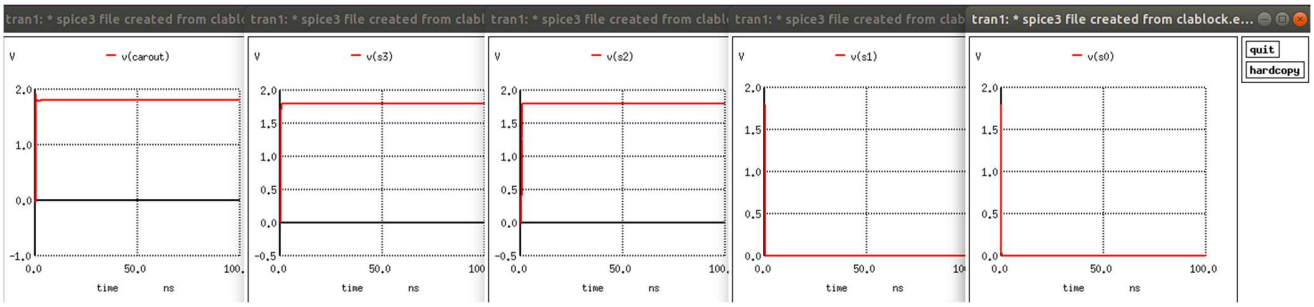
Comparison between prelayout and post layout values:

Let us take A=1111, B=1101. Since these numbers have more 1s, it will be clear if there is an error or not.

Prelayout:



Postlayout:



Question-7

Netlist for the full circuit:

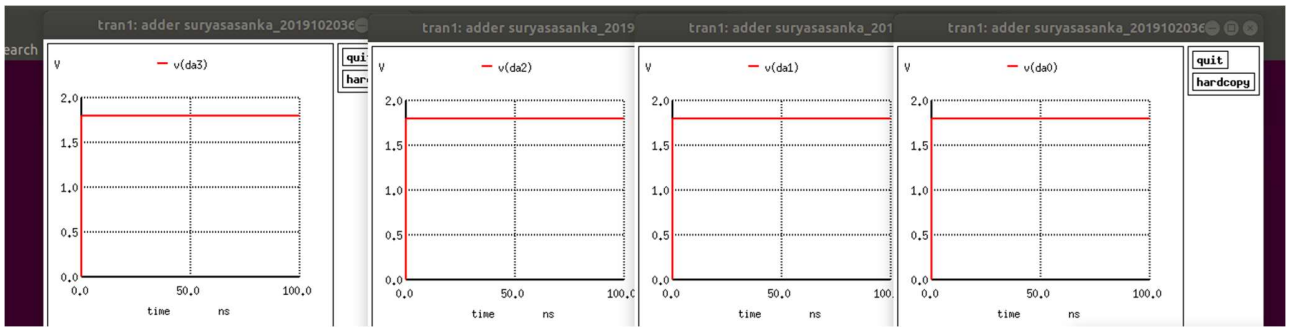
```
Adder Suryasasanka_2019102036
.include TSMC_180nm.txt
.param SUPPLY=1.8
.param LAMBDA=0.09u
.param width=10*LAMBDA
.global gnd vdd

VDD vdd gnd 'SUPPLY'
*vin_a DA0 0 pulse  1.8 0 0ns 100ps 100ps 100ns 200ns
vin_a DA0 0 pulse  0 1.8 0ns 100ps 100ps 100ns 200ns
*vin_a1 DA1 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_a1 DA1 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns
*vin_a2 DA2 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_a2 DA2 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns
*vin_a3 DA3 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_a3 DA3 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns

*vin_b DB0 0 pulse  1.8 0 0ns 100ps 100ps 100ns 200ns
vin_b DB0 0 pulse  0 1.8 0ns 100ps 100ps 100ns 200ns
*vin_b1 DB1 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_b1 DB1 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns
*vin_b2 DB2 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_b2 DB2 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns
vin_b3 DB3 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
*vin_b3 DB3 0 pulse 0 1.8 0ns 100ps 100ps 100ns 200ns



vin_cin C0 0 pulse 1.8 0 0ns 100ps 100ps 100ns 200ns
vin_clk clk 0 pulse 1.8 0 0ns 100ps 100ps 25ns 50ns


.subckt not y x
.param width_N={width}
.param width_P={2*width}

M1 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}

M2 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}
.ends not

.subckt and a b y
```

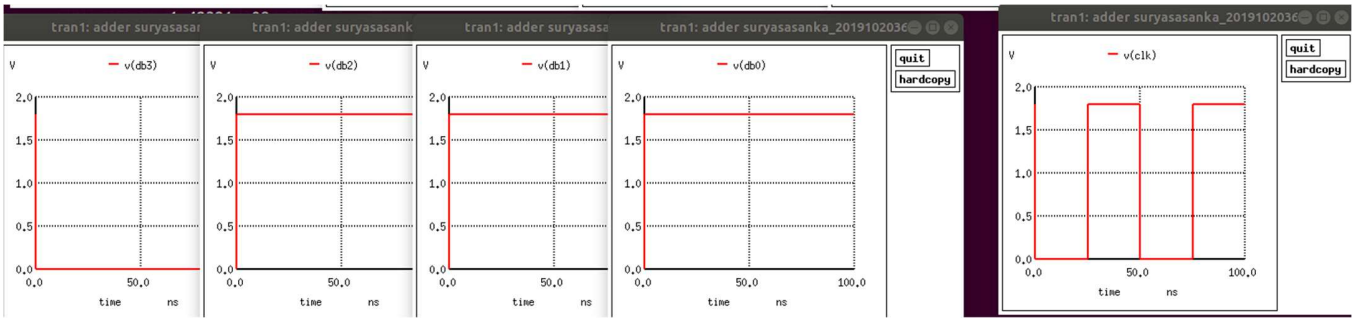
```
x1 A0 B0 P0 xor
x2 A1 B1 P1 xor
x3 A2 B2 P2 xor
x4 A3 B3 P3 xor

x5 A0 B0 G0 and
x6 A1 B1 G1 and
x7 A2 B2 G2 and
x8 A3 B3 G3 and
*-----------------------------CLA block

V_c1 C1 G0 '0'            $G0

x9 P1 G0 PG10 and         $P1G0
x10 G1 PG10 C2 or         $G1+P1G0

x11 P2 PG10 PPG210 and    $P2P1G0
x12 P2 G1 PG21 and        $P2G1
x13 PG21 PPG210 PGPPG or  $P2G1+P2P1G0
x14 G2 PGPPG C3 or        $C3=G2+P2G1+P2P1G0

x15 P3 PPG210 PPPG3210 and    $P3P2P1G0
x16 P3 PG21 PPG321 and        $P3P2G1
x17 PPPG3210 PPG321 last1 or  $P3P2G1+P3P2P1G0

x18 P3 G2 PG32 and            $P3G2
x19 G3 PG32 last2 or          $G3+P3G2

x20 last2 last1 Cout or       $G3+P3G2+P3P2G1+P3P2P1G0

*----------------Sum block

x21 P0 C0 S0 xor
x22 P1 C1 S1 xor
x23 P2 C2 S2 xor
x24 P3 C3 S3 xor

*----------------Output D-flipflop
x201 clk S0 Q0 Q0_bar d_ff
x202 clk S1 Q1 Q1_bar d_ff
x203 clk S2 Q2 Q2_bar d_ff
x204 clk S3 Q3 Q3_bar d_ff
x205 clk Cout Q4 Q4_bar d_ff
```

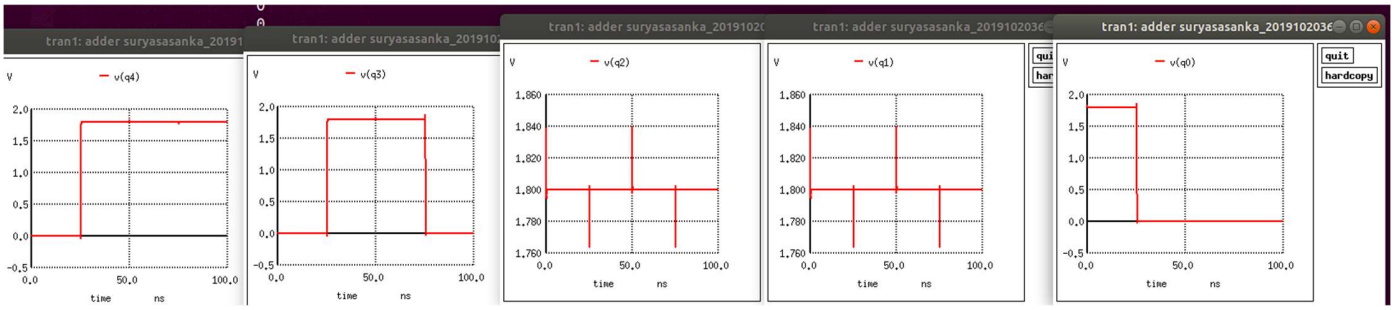
```
x21 P0 C0 S0 xor
x22 P1 C1 S1 xor
x23 P2 C2 S2 xor
x24 P3 C3 S3 xor

*----------------Output D-flipflop
x201 clk S0 Q0 Q0_bar d_ff
x202 clk S1 Q1 Q1_bar d_ff
x203 clk S2 Q2 Q2_bar d_ff
x204 clk S3 Q3 Q3_bar d_ff
x205 clk Cout Q4 Q4_bar d_ff

.tran 0.1n 100n



.control
set hcopypscolor = 1
set color0=white
set color1=black

run
plot v(clk)

plot v(DA0)
plot v(DA1)
plot v(DA2)
plot v(DA3)

plot v(DB0)
plot v(DB1)
plot v(DB2)
plot v(DB3)

plot v(Q0)
plot v(Q1)
plot v(Q2)
plot v(Q3)
plot v(Q4)

.endc
```

Plots:

We have taken DA=1111 DB=0111 which are the inputs to the input D-fipflops. The clock has time period 50ns with rising edge at 25ns and 75ns. The inputs to the D-flipflop enter the CLA at 25ns. The computation is done and is ready at the 5 output D-flipflops. At the next rising edge I.e. at 75ns, we get the values Q4 Q3 Q2 Q1 Q0 which is the calculated sum.







At 75ns, Q4=1 Q3=0 Q2=1 Q1=1 Q0=0 10110 which is the required sum of A+B.

Worst case delay of adder:

vin_a A0 0 pulse 1.8 0 0ns 100ps 100ps 50ns 100ns.

This is the input that I have taken for all the bits. Each of them changes from 0 to 1 at 50ns time. At that point, I have tried to calculate the delay which is the maximum due to the fact that every bit is 1, it results in the maximum number of computations.

I have used the .measure function as follows:

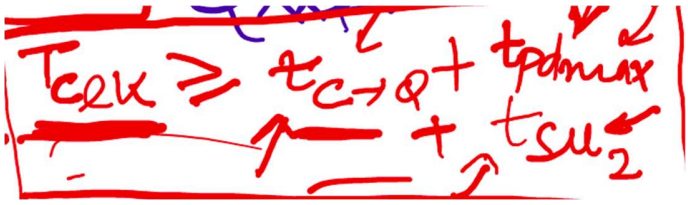
```
.measure tran tpdcout
+TRIG v(A0) val = 'SUPPLY/2' RISE = 1
+TARG v(Cout) val = 'SUPPLY/2' RISE = 1


.measure tran tpds3
+TRIG v(A0) val = 'SUPPLY/2' RISE = 1
+TARG v(S3) val = 'SUPPLY/2' RISE = 1
```

```
  Measurements for Transient Analysis

tpdcout              =  2.539482e-10 targ=  5.040395e-08 trig=  5.015000e-08
tpds3                =  3.295237e-10 targ=  5.047952e-08 trig=  5.015000e-08

ngspice 1 ->
```

Tpdmax=3.29 x 10^-10 which is the maximum delay.

Calculating the maximum clock frequency:

In the class we have derived the formula for maximum clock frequency as follows



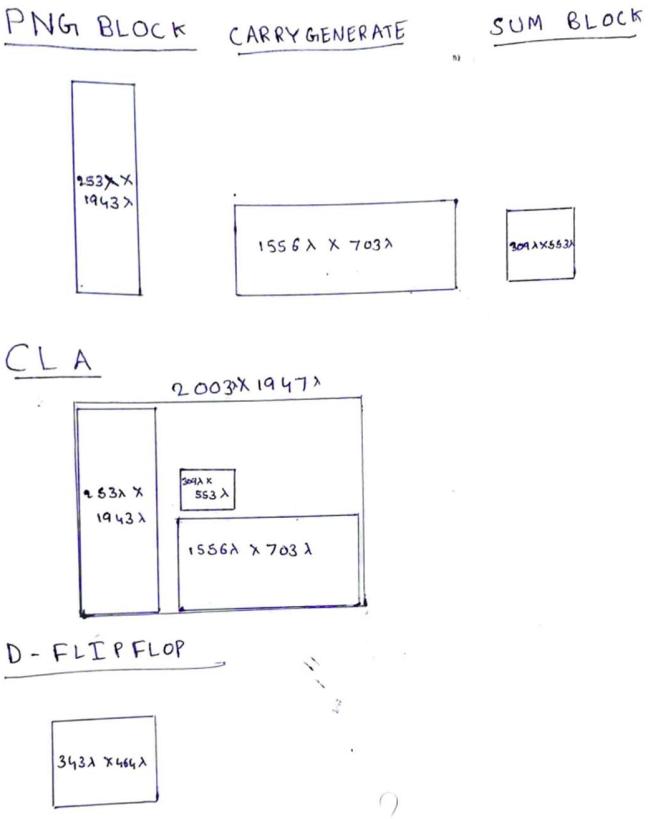
Tclk>= TclkQ+ Tpdmax+ Tsu.

We have already found out Tclk-Q=1.477 x10^-10, Tsu= 0.14ns

Tclkmin=6.617 x 10^-10

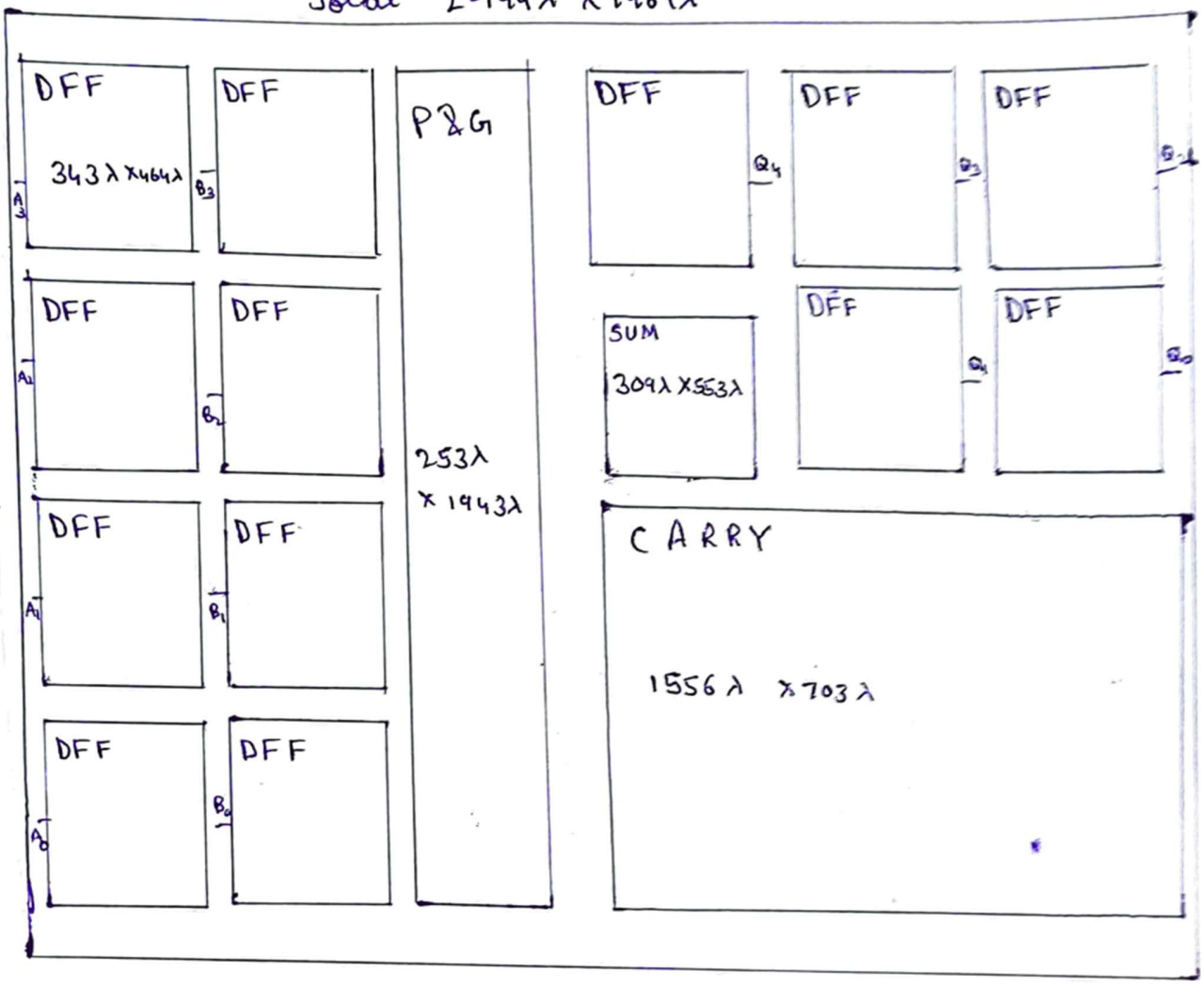
Fclkmax=1.511 x 10^9=1.511 GHz

Question-8


PNG BLOCK
CARRY GENERATE
SUM BLOCK
253λ X 1943λ
1556λ X 703λ
309λ X 553λ
CLA
2003λ X 1947λ
253λ X 1943λ
309λ X 553λ
1556λ X 703λ
D - FLIP FLOP
343λ X 466λ

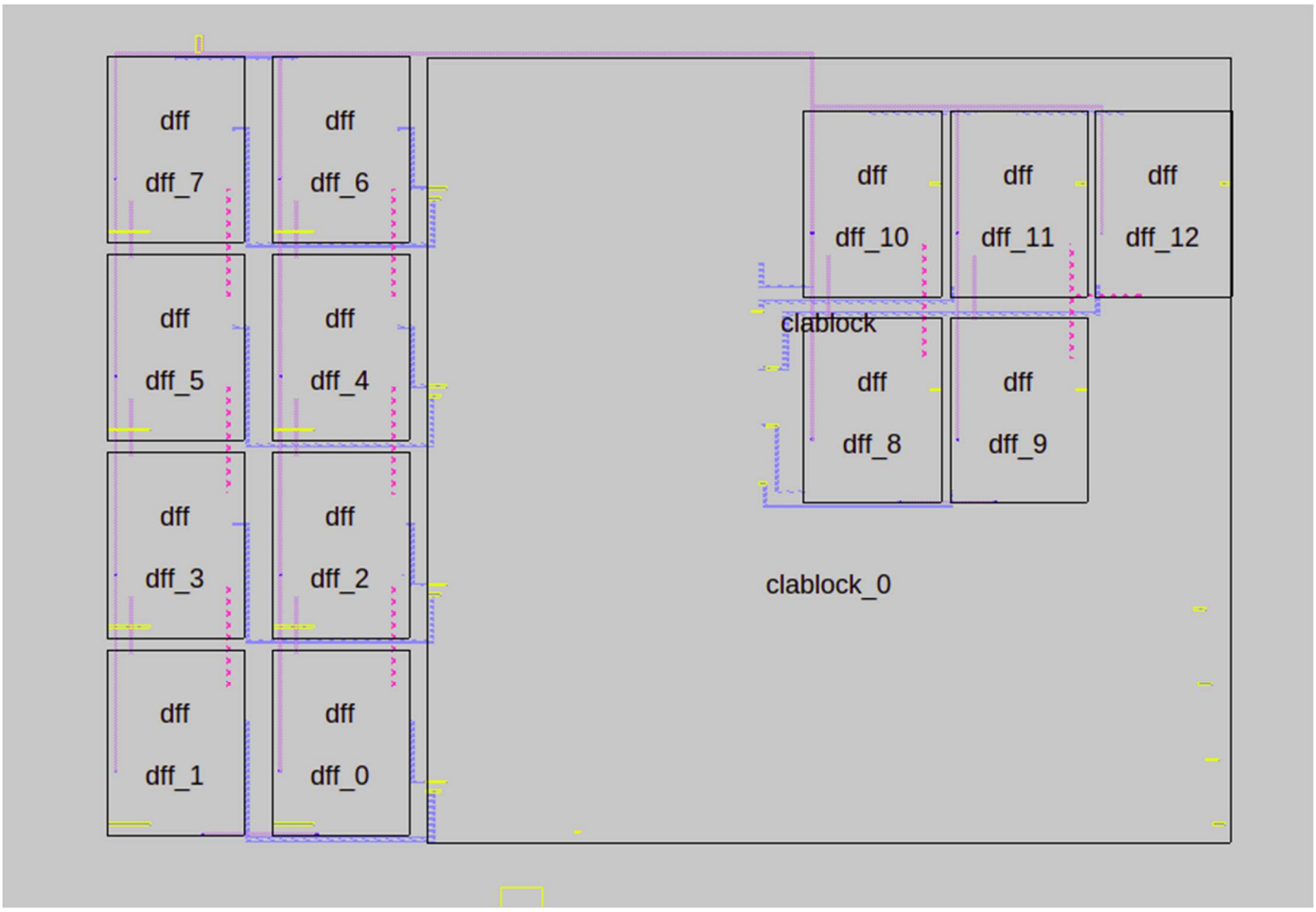
FLOOR PLAN

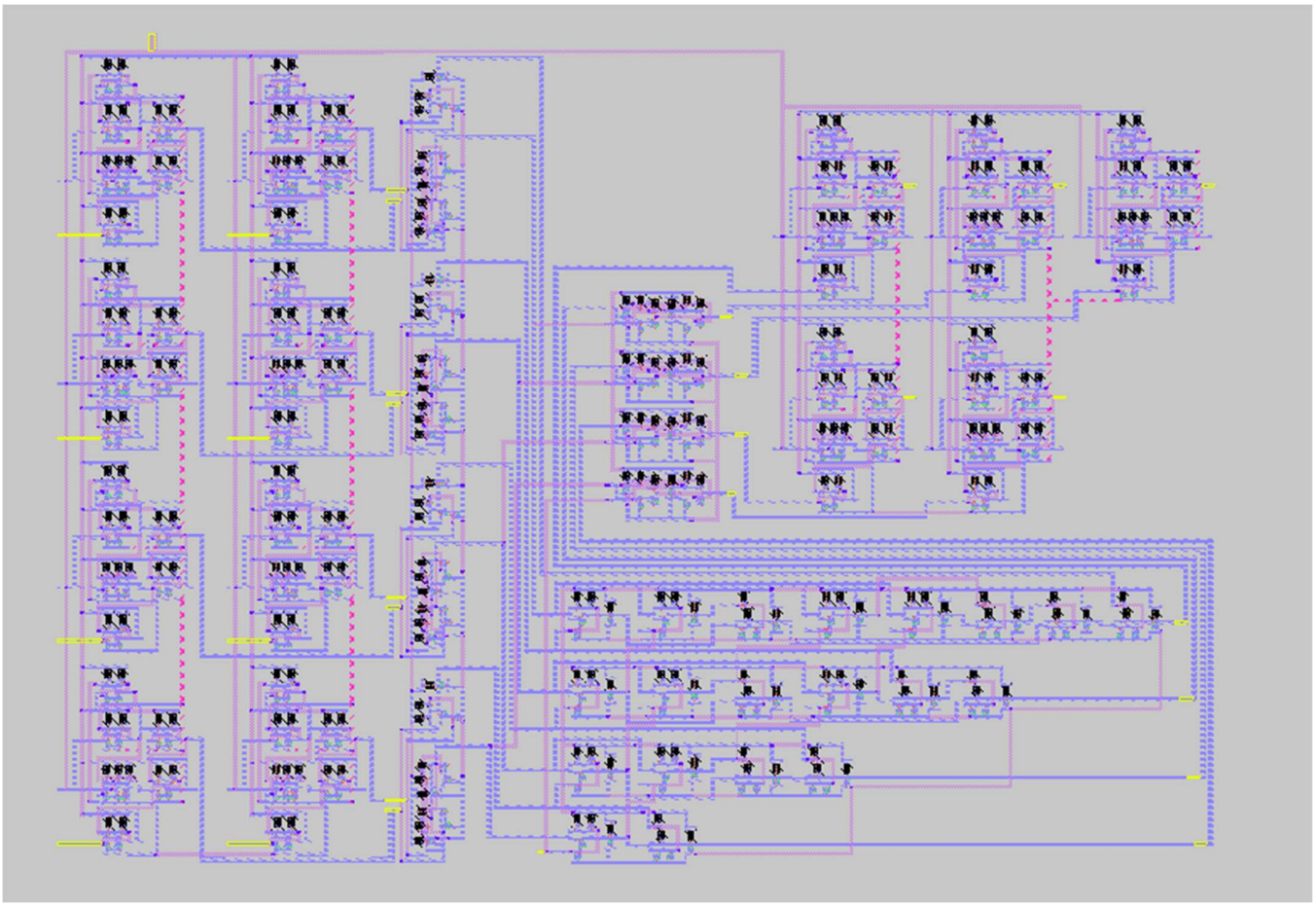
Total 2799λ x 1961λ



Question-9, 10

Final layout:





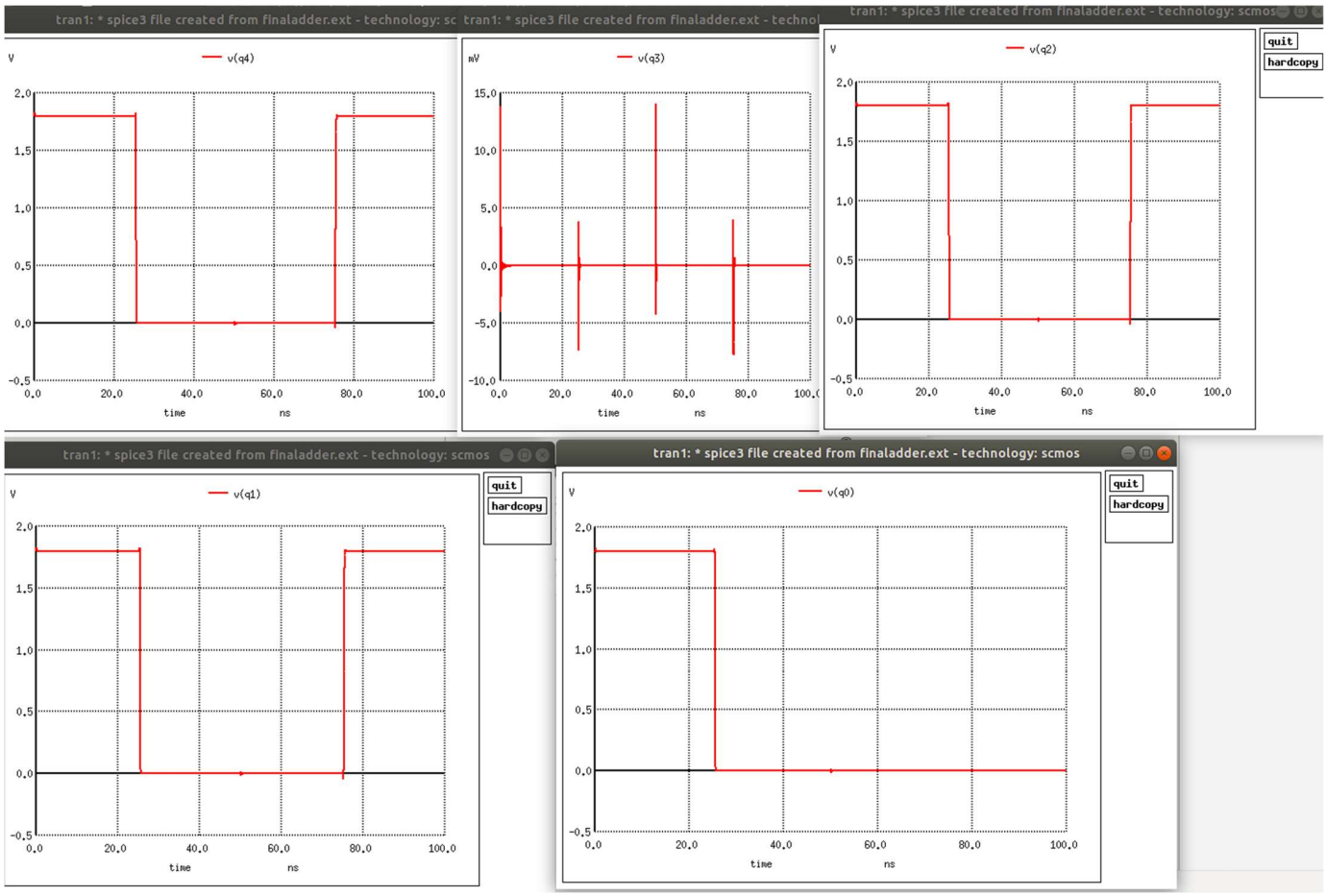
Extracted netlists:

The final extracted netlist has 578 MOSFETs and 2186 capacitors

```
+  ad=120 pd=64 as=0 ps=0
M1565 dff_7/m1_n49_n87# dff_7/m1_n123_103# dff_7/nand3_0/a_106_9# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1566 A3 dff_7/m1_n123_103# vdd dff_7/nandgate_0/w_122_92# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1567 A3 dff_7/m1_0_n20# vdd dff_7/nandgate_0/w_122_92# CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1568 dff_7/nandgate_0/a_137_45# dff_7/m1_0_n20# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1569 A3 dff_7/m1_n123_103# dff_7/nandgate_0/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1570 dff_7/m1_n123_103# dff_7/m1_n114_50# vdd dff_7/nandgate_3/w_122_92# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1571 dff_7/m1_n123_103# clk vdd dff_7/nandgate_3/w_122_92# CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1572 dff_7/nandgate_3/a_137_45# clk gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1573 dff_7/m1_n123_103# dff_7/m1_n114_50# dff_7/nandgate_3/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
M1574 dff_7/m1_n114_50# dff_7/m1_n140_n124# vdd dff_7/nandgate_4/w_122_92# CMOSP w=20 l=2
+  ad=240 pd=104 as=0 ps=0
M1575 dff_7/m1_n114_50# dff_7/m1_n123_103# vdd dff_7/nandgate_4/w_122_92# CMOSP w=20 l=2
+  ad=0 pd=0 as=0 ps=0
M1576 dff_7/nandgate_4/a_137_45# dff_7/m1_n123_103# gnd gnd CMOSN w=10 l=2
+  ad=120 pd=64 as=0 ps=0
M1577 dff_7/m1_n114_50# dff_7/m1_n140_n124# dff_7/nandgate_4/a_137_45# gnd CMOSN w=10 l=2
+  ad=60 pd=32 as=0 ps=0
C0 clablock_0/carrygen_0/m1_174_152# gnd 1.0fF
C1 clablock_0/m1_248_764# clablock_0/m1_253_1444# 0.2fF
C2 vdd clablock_0/png_0/xorgate_1/a_56_44# 0.2fF
C3 clablock_0/sumblock_0/xorgate_3/inverter_0/w_n13_n7# clablock_0/sumblock_0/xorgate_3/a_48_n7# 0.0fF
C4 dff_12/nandgate_3/a_137_45# gnd 0.2fF
C5 S2 clk 0.4fF
C6 dff_8/m1_0_n20# vdd 0.9fF
C7 gnd clk 0.3fF
C8 dff_0/m1_n114_50# clk 0.1fF
C9 clablock_0/carrygen_0/andgate_7/a_n61_61# clablock_0/carrygen_0/andgate_7/inverter_0/w_n13_n7# 0.1fF
C10 clablock_0/carrygen_0/orgate_0/w_n74_71# clablock_0/m1_235_462# 0.1fF
C11 vdd dff_7/nandgate_3/w_122_92# 0.2fF
C12 clk dff_7/m1_n140_n124# 0.6fF
C13 dff_3/nandgate_4/w_122_92# dff_3/m1_n140_n124# 0.1fF
C14 gnd clablock_0/m1_253_1444# 0.1fF
C15 clablock_0/carrygen_0/orgate_4/a_n63_n10# clablock_0/carrygen_0/m1_777_387# 0.1fF
C16 clablock_0/carrygen_0/m1_174_38# clablock_0/carrygen_0/m1_174_152# 0.1fF
```

```
C2148 clablock_0/carrygen_0/m1_567_341# gnd 1.7fF
C2149 clablock_0/carrygen_0/orgate_5/a_n59_77# gnd 0.0fF
C2150 clablock_0/carrygen_0/m1_947_392# gnd 2.1fF
C2151 clablock_0/carrygen_0/orgate_5/w_n65_31# gnd 0.9fF
C2152 clablock_0/carrygen_0/orgate_5/w_n74_71# gnd 0.9fF
C2153 clablock_0/carrygen_0/orgate_5/a_n63_n10# gnd 0.7fF
C2154 clablock_0/carrygen_0/orgate_5/inverter_0/w_n13_n7# gnd 0.9fF
C2155 clablock_0/carrygen_0/andgate_1/a_n58_n25# gnd 0.1fF
C2156 clablock_0/carrygen_0/andgate_1/w_n42_50# gnd 1.0fF
C2157 clablock_0/carrygen_0/andgate_1/w_n76_50# gnd 1.0fF
C2158 clablock_0/carrygen_0/andgate_1/a_n61_61# gnd 0.6fF
C2159 clablock_0/carrygen_0/andgate_1/inverter_0/w_n13_n7# gnd 0.9fF
C2160 clablock_0/carrygen_0/andgate_2/a_n58_n25# gnd 0.1fF
C2161 clablock_0/carrygen_0/m1_174_38# gnd 4.2fF
C2162 clablock_0/carrygen_0/andgate_2/w_n42_50# gnd 1.0fF
C2163 clablock_0/carrygen_0/andgate_2/w_n76_50# gnd 1.0fF
C2164 clablock_0/carrygen_0/andgate_2/a_n61_61# gnd 0.6fF
C2165 clablock_0/carrygen_0/andgate_2/inverter_0/w_n13_n7# gnd 0.9fF
C2166 clablock_0/carrygen_0/orgate_1/a_n59_77# gnd 0.0fF
C2167 clablock_0/carrygen_0/orgate_1/w_n65_31# gnd 0.9fF
C2168 clablock_0/carrygen_0/orgate_1/w_n74_71# gnd 0.9fF
C2169 clablock_0/carrygen_0/orgate_1/a_n63_n10# gnd 0.6fF
C2170 clablock_0/carrygen_0/orgate_1/inverter_0/w_n13_n7# gnd 0.9fF
C2171 clablock_0/carrygen_0/m1_567_199# gnd 0.7fF
C2172 clablock_0/carrygen_0/orgate_2/a_n59_77# gnd 0.0fF
C2173 clablock_0/carrygen_0/orgate_2/w_n65_31# gnd 0.9fF
C2174 clablock_0/carrygen_0/orgate_2/w_n74_71# gnd 0.9fF
C2175 clablock_0/carrygen_0/orgate_2/a_n63_n10# gnd 0.7fF
C2176 clablock_0/carrygen_0/orgate_2/inverter_0/w_n13_n7# gnd 0.9fF
C2177 clablock_0/carrygen_0/andgate_0/a_n58_n25# gnd 0.3fF
C2178 clablock_0/carrygen_0/andgate_0/w_n42_50# gnd 1.1fF
C2179 clablock_0/carrygen_0/andgate_0/w_n76_50# gnd 1.0fF
C2180 clablock_0/carrygen_0/andgate_0/a_n61_61# gnd 0.9fF
C2181 clablock_0/carrygen_0/andgate_0/inverter_0/w_n13_n7# gnd 0.9fF
C2182 clablock_0/carrygen_0/orgate_0/a_n59_77# gnd 0.0fF
C2183 clablock_0/carrygen_0/orgate_0/w_n65_31# gnd 0.9fF
C2184 clablock_0/carrygen_0/orgate_0/w_n74_71# gnd 0.9fF
C2185 clablock_0/carrygen_0/orgate_0/a_n63_n10# gnd 0.7fF
C2186 clablock_0/carrygen_0/orgate_0/inverter_0/w_n13_n7# gnd 0.9fF

.tran 0.1n 100n
```

Let us run this circuit for the same input of prelayout simulation I.e. A=1111 B=0111



At rising edge of the clock at 25ns, the input values enter through the D-flip flop and operations on them are performed. At 75ns which is the next rising edge of the clock, the calculated outputs appear as Q4, Q3, Q2, Q1, Q0.

The output is 10110 which is the correct answer.

There is a link attached in the comments containing my netlists and layouts. The file named finaladder.spice in newlayouts folder is the

netlist which gives these outputs. We can change the inputs there to verify for further values.

To calculate the post layout delay of the CLA, we use add the .measure function to the extracted netlist to get the maximum delay.

```
C1060 carrygen_0/andgate_0/w_n76_50# gnd 1.0fF
C1061 carrygen_0/andgate_0/a_n61_61# gnd 0.9fF
C1062 carrygen_0/andgate_0/inverter_0/w_n13_n7# gnd 0.9fF
C1063 carrygen_0/orgate_0/a_n59_77# gnd 0.0fF
C1064 carrygen_0/orgate_0/w_n65_31# gnd 0.9fF
C1065 carrygen_0/orgate_0/w_n74_71# gnd 0.9fF
C1066 carrygen_0/orgate_0/a_n63_n10# gnd 0.7fF
C1067 carrygen_0/orgate_0/inverter_0/w_n13_n7# gnd 0.9fF
.tran 0.1n 100n

.measure tran tpdcar4
+TRIG v(A0) val = 'SUPPLY/2' RISE = 1
+TARG v(Carout) val = 'SUPPLY/2' RISE = 1

.measure tran tpds3
+TRIG v(A0) val = 'SUPPLY/2' RISE = 1
+TARG v(S3) val = 'SUPPLY/2' RISE = 1
```

```
 Measurements for Transient Analysis

tpdcar4            =  2.353523e-10 targ=  5.038535e-08 trig=  5.015000e-08
tpds3              =  4.896761e-10 targ=  5.063968e-08 trig=  5.015000e-08

ngspice 1 ->
```

Tpdmax=tpds3=4.896 x 10^-10

Tclk>= TclkQ+ Tpdmax+ Tsu.

We have already found out Tclk-Q=2.22 x10^-10, Tsu= 0.16ns

Tclkmin=8.716 x 10^-10 s

Fclkmax=1.14 x 10^9 Hz= 1.14GHz

	Tclk-Q	Tsu	Thold	Tpdmax	Tclkmin	Fclkmax
Prelayout	0.147ns	0.14ns	0.095ns	0.329ns	0.6617ns	1.511GHz
Postlayout	0.222ns	0.16ns	0.05ns	0.4896ns	0.8716ns	1.14GHz

The maximum clock frequency for which my extracted layout works properly is 1.14GHz

Question-11

This is a very simple part where we write Verilog modules for adding two numbers. It has two parts. Structural part and data path. I have implemented the structural path directly in the data path. The following are the modules.

```
`timescale 1ns/1ps

module adder(clock, a, b, sum);

input clock;
input [3:0] a, b;
output reg [4:0]sum;

reg [3:0] p, g;
reg [4:0] carry;
initial carry[0]=0;

always @(posedge clock)
begin
        //This is the structural part. We can write this also in another
        //module but since our goal is to verify simulations, we use directly
        p=a^b;
        g=a&b;

        carry[1]=g[0];
        carry[2]=(g[1]) | (p[1] & g[0]);
        carry[3]=(g[2]) | (p[2] & g[1]) | (p[2] & p[1] & p[0]);
        carry[4]=(g[3]) | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3]&p[2]&p[1]&g[0]);

        sum[0]=p[0]^carry[0];
        sum[1]=p[1]^carry[1];
        sum[2]=p[2]^carry[2];
        sum[3]=p[3]^carry[3];
        sum[4]=carry[4];


end


endmodule
~
~
```

```
`timescale 1ns / 1ps
module adder_tb;

reg [3:0] a,b;
reg clk;
wire [4:0] sum;

adder uut (clk, a, b, sum);

initial begin
        $dumpfile ("adder_o.vcd");
        $dumpvars (0,adder_tb);
        clk = 0;
        #10;
        a=3;
        b=4;
        #10;

        a=5;
        b=7;
        #10;

//      a=8;
//      b=2;
        $finish;
end
always #5  clk = ~clk ;

endmodule
~
```