VLSI Project

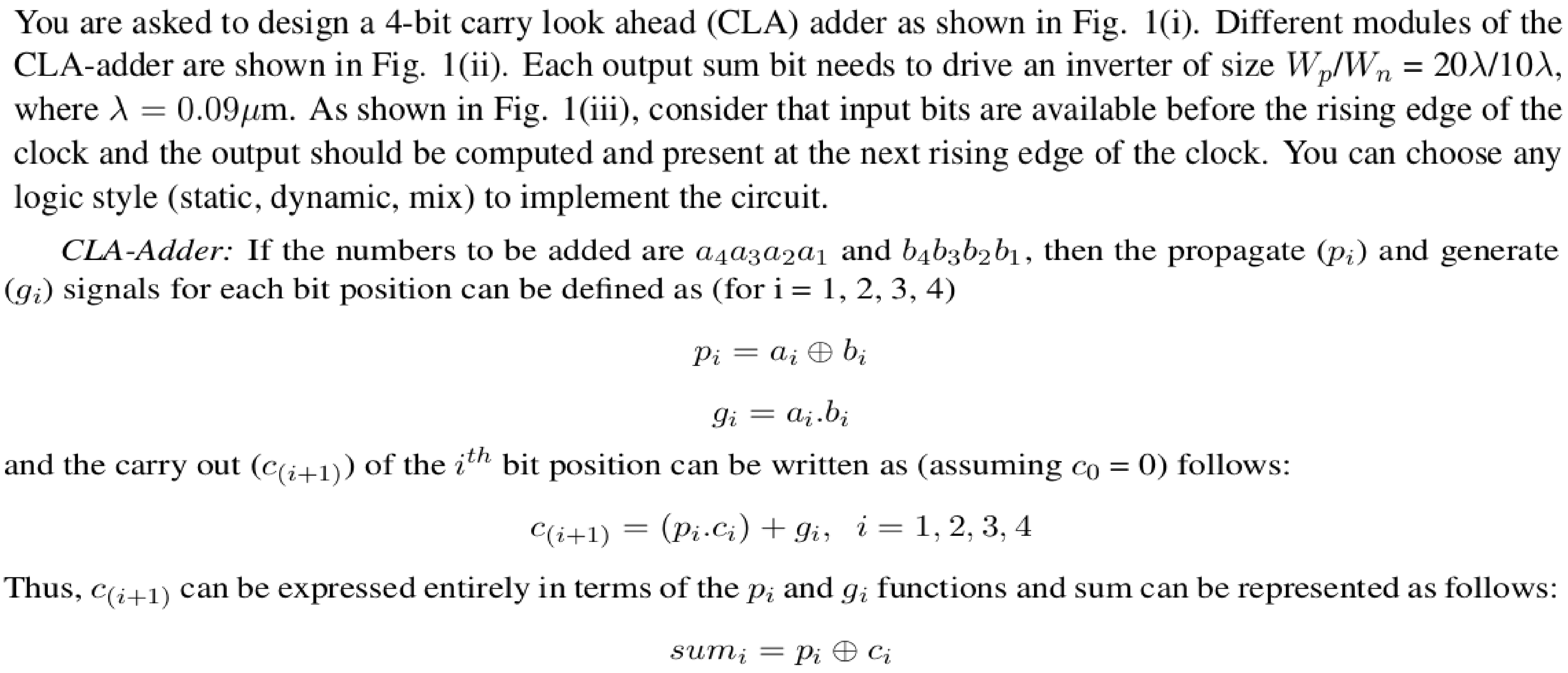
Viswanadh

2019112011

General Points

- All subparts have been answered accordingly.
- Few explanations were given at last for questions instead of mentioning in respective subparts to maintain a flow
- Plot titles couldn’t be attached to all plots hence I didn’t crop the title bar to include the title of the netlist.
- Supporting notes have been written and attached instead of typing out few equations and other information
- Codes and supporting files can be found here

4-bit carry look ahead (CLA)



You are asked to design a 4-bit carry look ahead (CLA) adder as shown in Fig. 1(i). Different modules of the CLA-adder are shown in Fig. 1(ii). Each output sum bit needs to drive an inverter of size $W_p/W_n = 20\lambda/10\lambda$, where $\lambda = 0.09\mu$m. As shown in Fig. 1(iii), consider that input bits are available before the rising edge of the clock and the output should be computed and present at the next rising edge of the clock. You can choose any logic style (static, dynamic, mix) to implement the circuit.

CLA-Adder: If the numbers to be added are $a_4a_3a_2a_1$ and $b_4b_3b_2b_1$, then the propagate (p_i) and generate (g_i) signals for each bit position can be defined as (for i = 1, 2, 3, 4)

$$p_i = a_i \oplus b_i$$

$$g_i = a_i.b_i$$

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

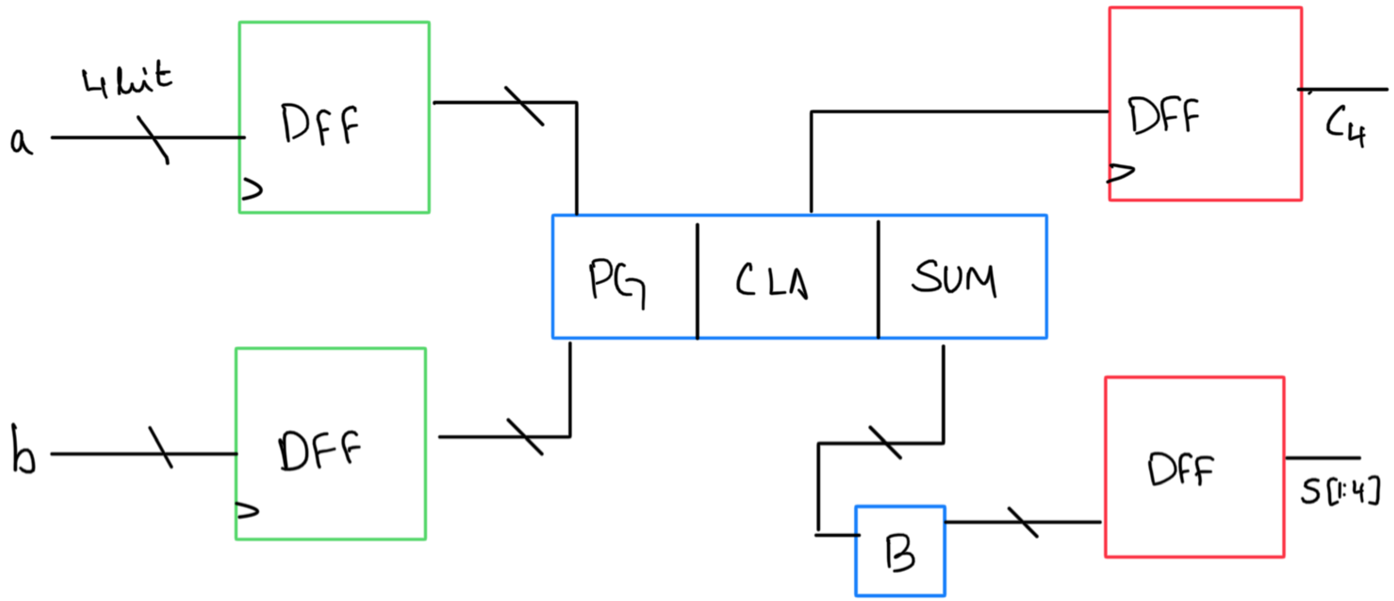
$$c_{(i+1)} = (p_i.c_i) + g_i, \quad i = 1, 2, 3, 4$$

Thus, $c_{(i+1)}$ can be expressed entirely in terms of the p_i and g_i functions and sum can be represented as follows:

$$sum_i = p_i \oplus c_i$$

Question 1

Briefly discuss your proposed structure for the adder.



----- => Input Block

----- => Intermediate Block

----- => Output Block

- *DFF* : D- flip flop with positive edge trigger.
- *PG* : Finds AND and XOR of given inputs.
- *CLA*: Finds the carry of a bit without propagation delay (look ahead)
- *SUM*: Performs XOR of given inputs.
- *B*: (BUFFer)Logically maps the input to HIGH(>0.6V~) or LOW (<0.6 ~)

- Here, all operations performed are bitwise operations.
- The above blocks mentioned perform similar operations

For my structure, initially the two 4-bit numbers, a and b, are passed on to 2 DFFs respectively. From here the outputs of the flip flops are passed on to main adder circuit which mainly has three sub blocks: PG, CLA and SUM block connected in series respectively. Then outputs of the SUM block is passed into a BUFFer block to logically map the values to corresponding LOW or HIGH. These (sum) values are finally stored in the flip flops and the output carry (C4) which is obtained from CLA block is also stored in a separate flip flop. Most of the gates are implemented using PTL gates barring few. This is explicitly mentioned in coming questions.

Question 2 and 3

Give design details (topology and sizing) of each block (D-flip-flop, adder modules).Simulate each block and verify its functionality using NGSPICE.

Firstly, blocks present are:

- *PG*
- *CLA*
- *SUM*
- *DFF*

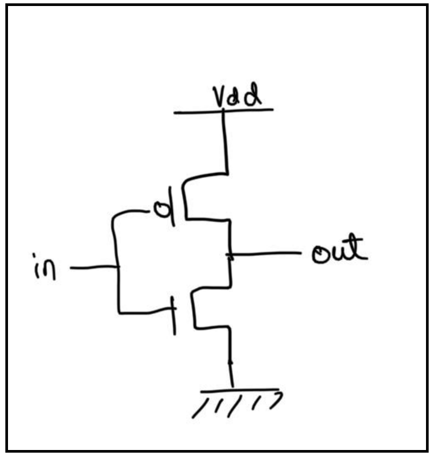
Logic Gates

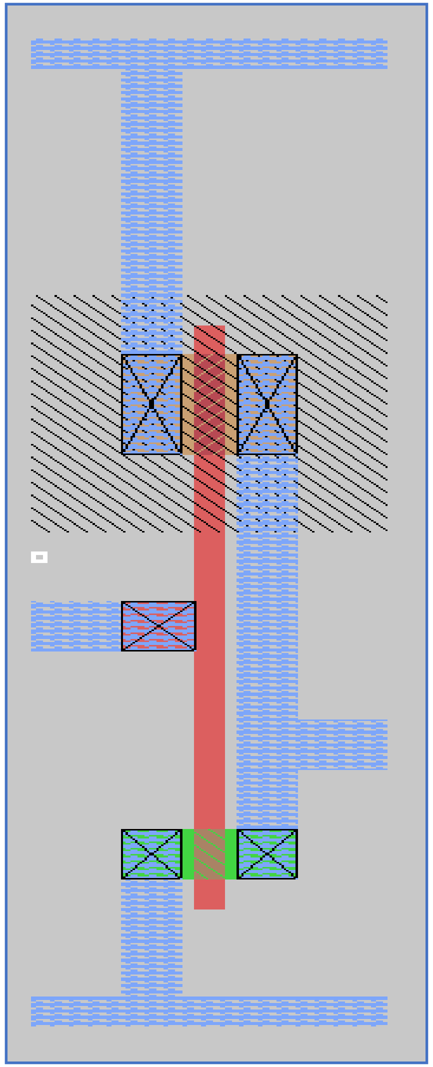
Before we dive into each block, we would first go through logic gates upon which these blocks are built upon.

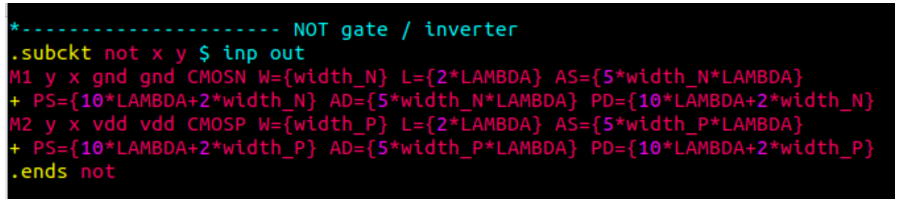
- NOT Gate
- AND Gate
- OR Gate
- XOR Gate
- BUFFER
- NAND Gate (2 and 3 input)

- Above gates are all built using Pass transistor logic except for inverter, buffer, NAND which are built using CMOS logic.
- PTL is used as less transistors would be used which is very useful here as already our circuit avoids propagation delay trading off with size of the circuit.
- But there are variations in logical HIGH obtained for which BUFFers are used to map these values to VDD and gnd.

NOT Gate





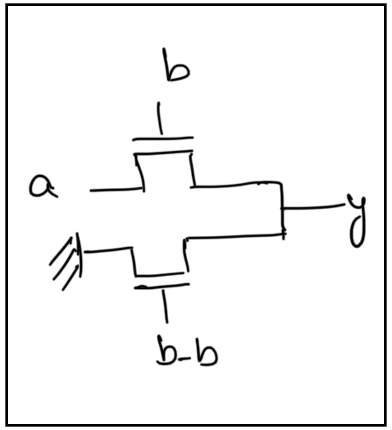


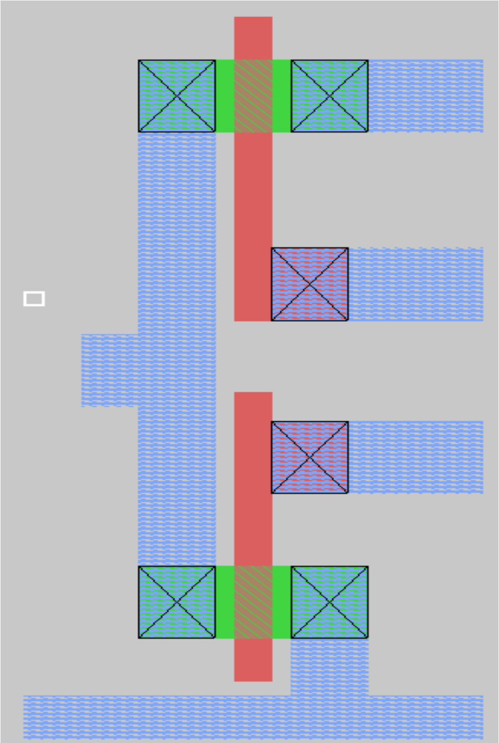
```
*---------------------- NOT gate / inverter
.subckt not x y $ inp out
M1 y x gnd gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 y x vdd vdd CMOSP W={width_P} L={2*LAMBDA} AS={5*width_P*LAMBDA}
+ PS={10*LAMBDA+2*width_P} AD={5*width_P*LAMBDA} PD={10*LAMBDA+2*width_P}
.ends not
```

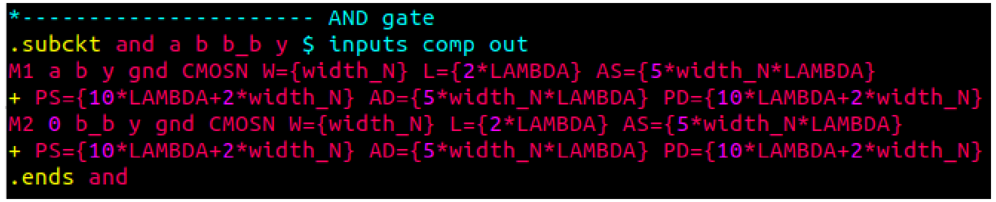
Width of NMOS = 5λ

Width of PMOS = 10λ

AND Gate



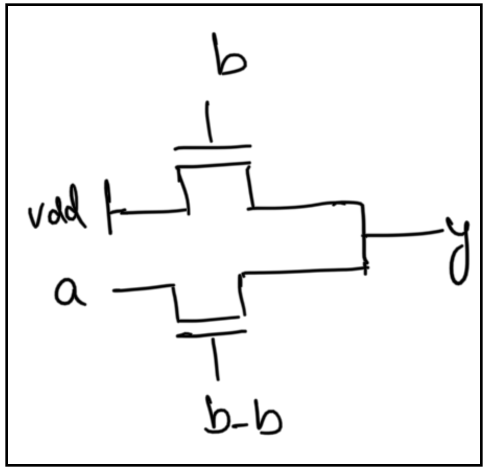


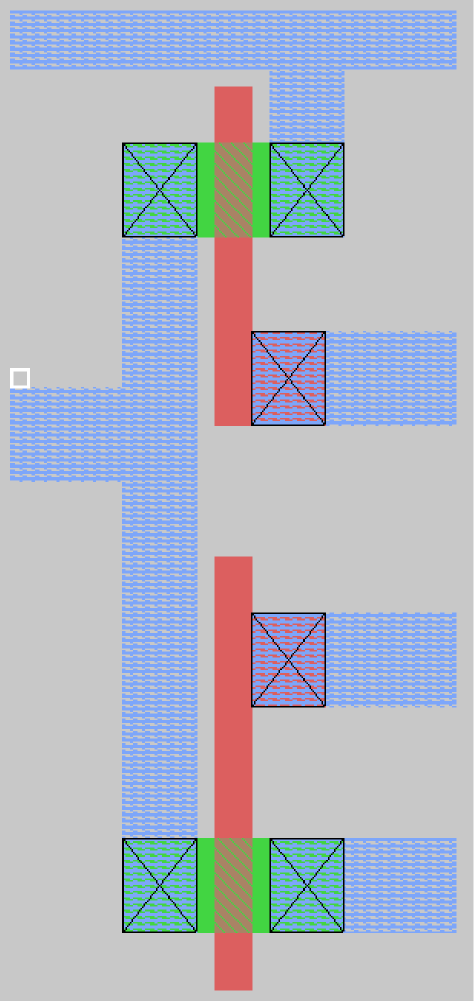


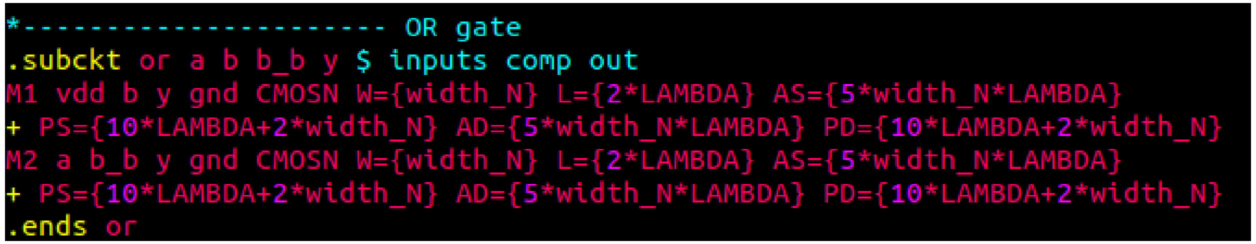
```
*---------------------- AND gate
.subckt and a b b_b y $ inputs comp out
M1 a b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 0 b_b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends and
```

Width of NMOS = 5λ
Width of PMOS = 10λ

OR Gate



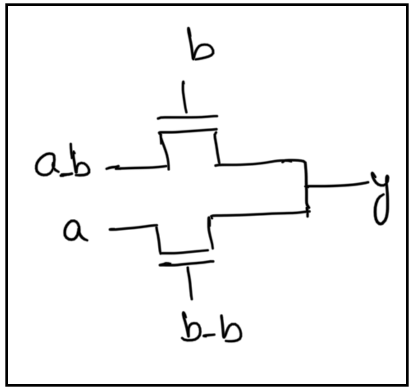


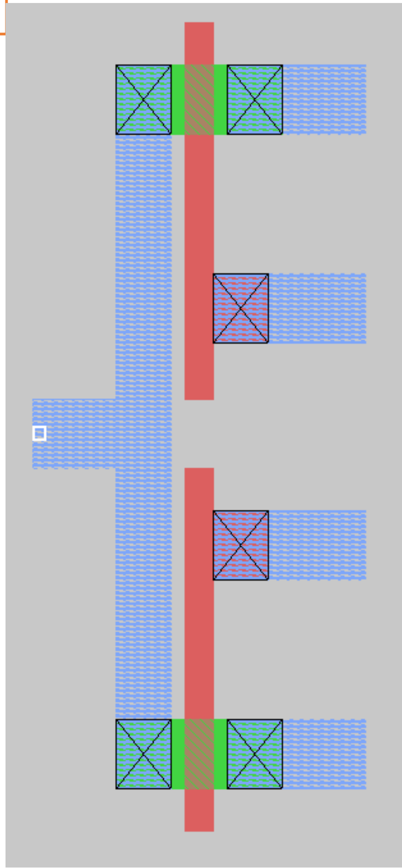


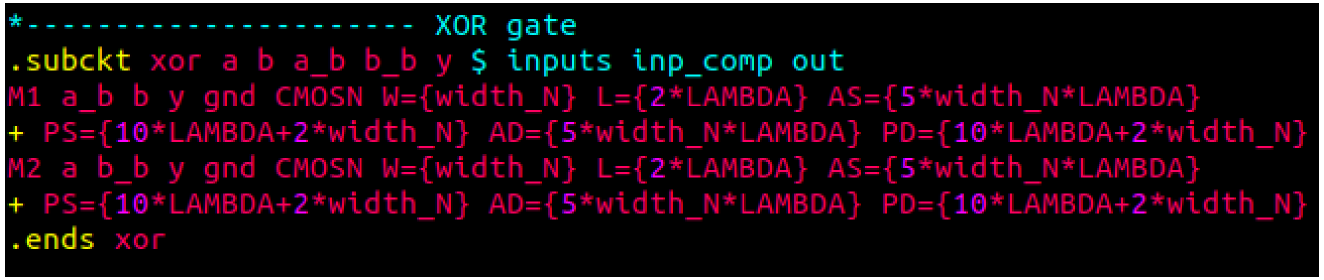
```
*---------------------- OR gate
.subckt or a b b_b y $ inputs comp out
M1 vdd b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 a b_b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends or
```

Width of NMOS = 5λ
Width of PMOS = 10λ

XOR Gate



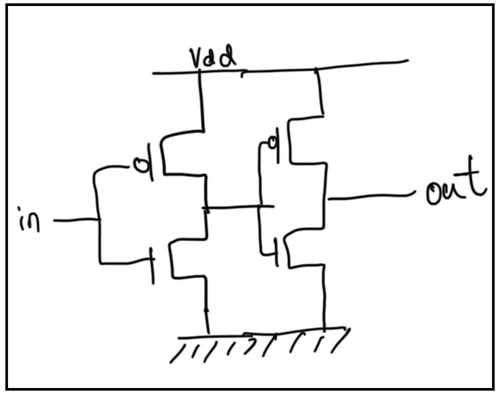


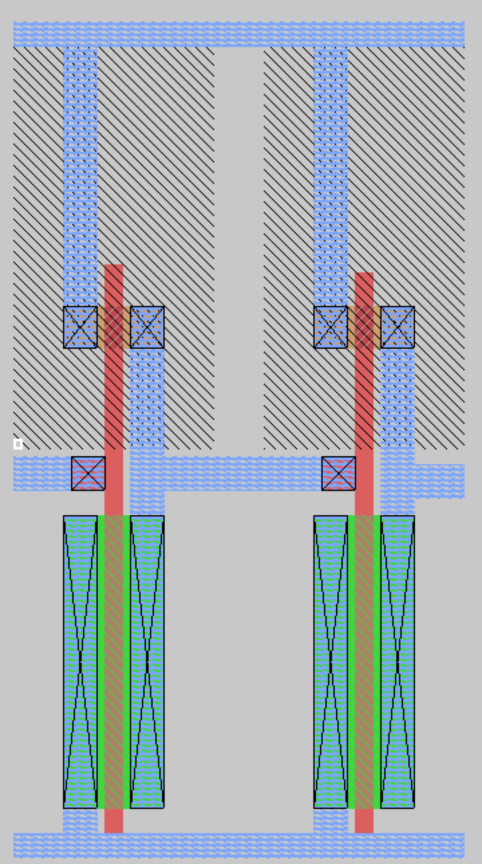


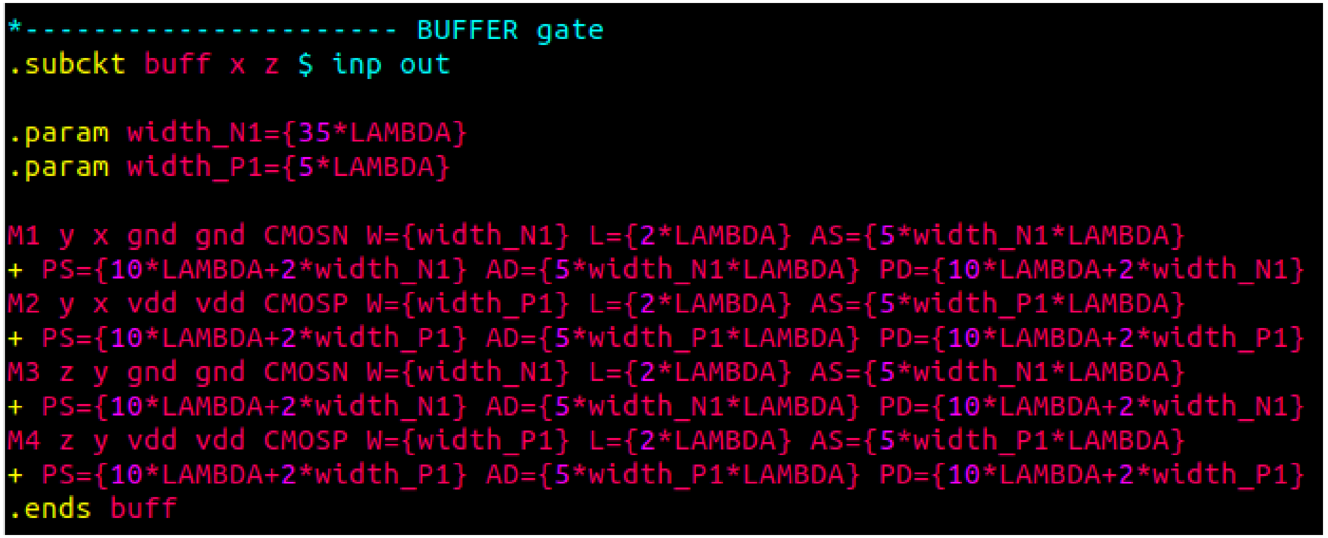
```
*----------------------- XOR gate
.subckt xor a b a_b b_b y $ inputs inp_comp out
M1 a_b b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 a b_b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
.ends xor
```

Width of NMOS = 5λ
Width of PMOS = 10λ

BUFFER



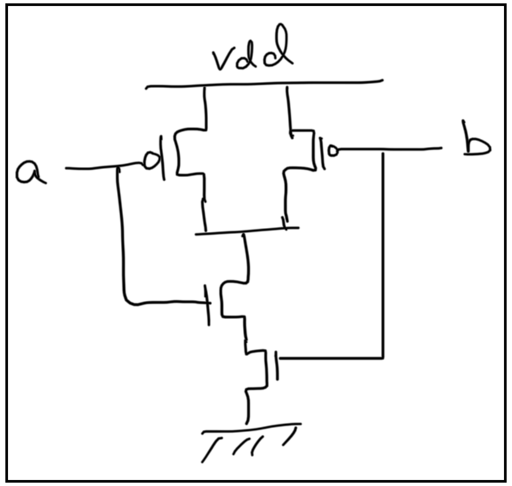


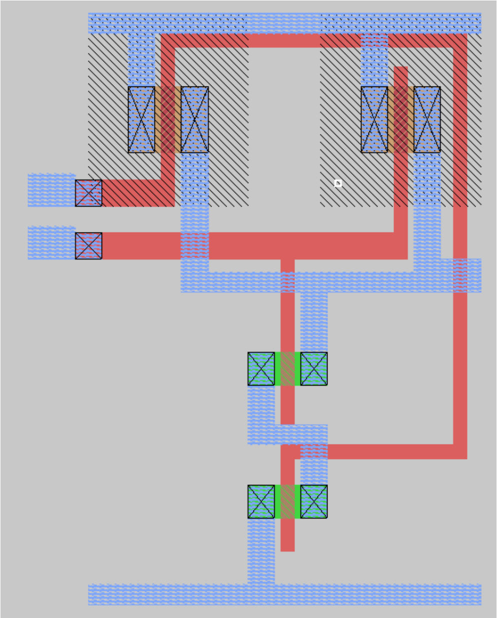


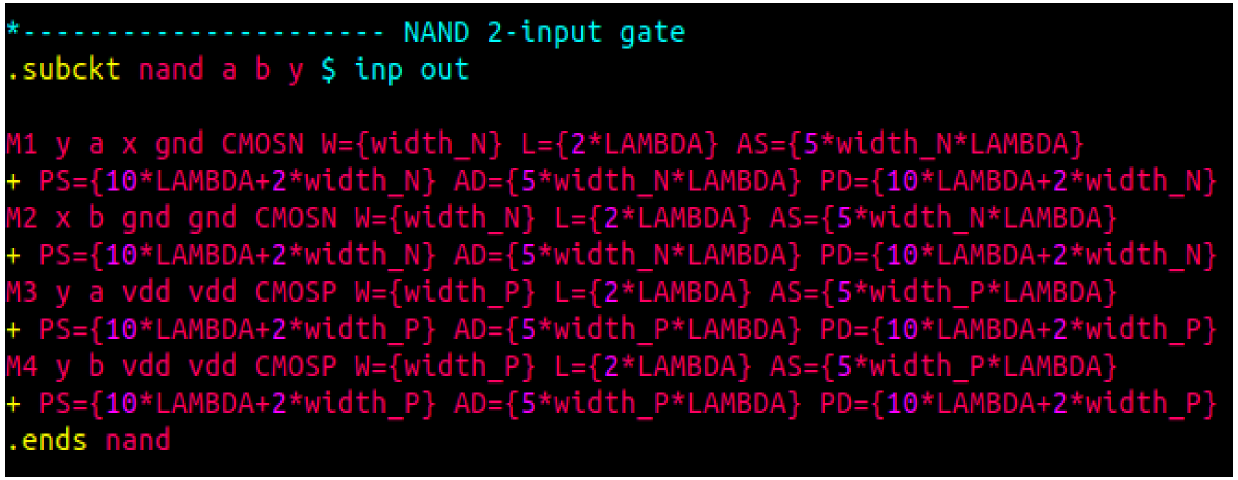
```
*---------------------- BUFFER gate
.subckt buff x z $ inp out

.param width_N1={35*LAMBDA}
.param width_P1={5*LAMBDA}

M1 y x gnd gnd CMOSN W={width_N1} L={2*LAMBDA} AS={5*width_N1*LAMBDA}
+ PS={10*LAMBDA+2*width_N1} AD={5*width_N1*LAMBDA} PD={10*LAMBDA+2*width_N1}
M2 y x vdd vdd CMOSP W={width_P1} L={2*LAMBDA} AS={5*width_P1*LAMBDA}
+ PS={10*LAMBDA+2*width_P1} AD={5*width_P1*LAMBDA} PD={10*LAMBDA+2*width_P1}
M3 z y gnd gnd CMOSN W={width_N1} L={2*LAMBDA} AS={5*width_N1*LAMBDA}
+ PS={10*LAMBDA+2*width_N1} AD={5*width_N1*LAMBDA} PD={10*LAMBDA+2*width_N1}
M4 z y vdd vdd CMOSP W={width_P1} L={2*LAMBDA} AS={5*width_P1*LAMBDA}
+ PS={10*LAMBDA+2*width_P1} AD={5*width_P1*LAMBDA} PD={10*LAMBDA+2*width_P1}
.ends buff
```

Width of NMOS = 35λ
Width of PMOS = 5λ

NAND Gate





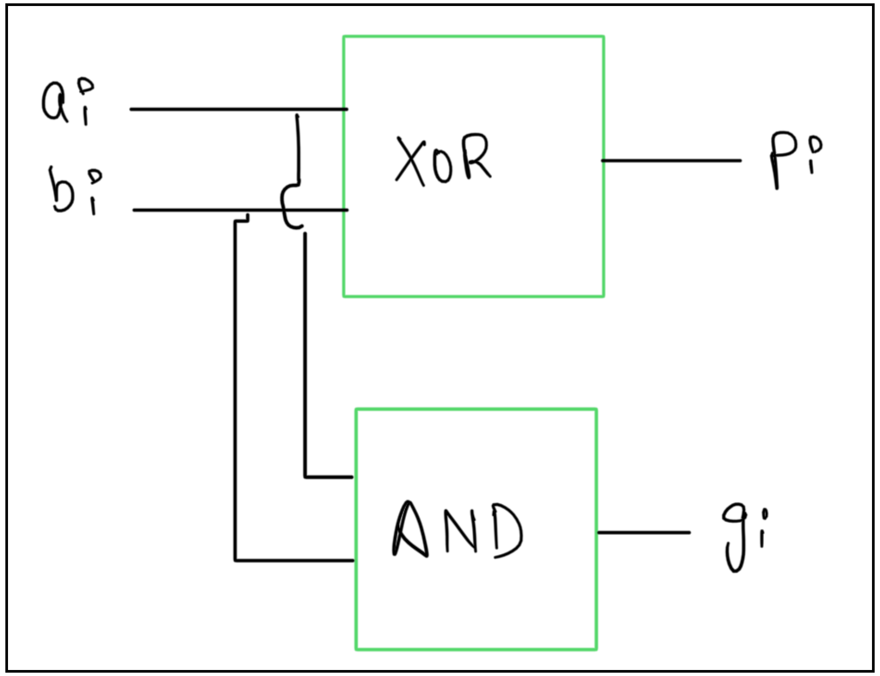


```
*---------------------- NAND 2-input gate
.subckt nand a b y $ inp out

M1 y a x gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 x 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}
M3 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}
M4 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}
.ends nand
```

Width of NMOS = 5λ
Width of PMOS = 10λ

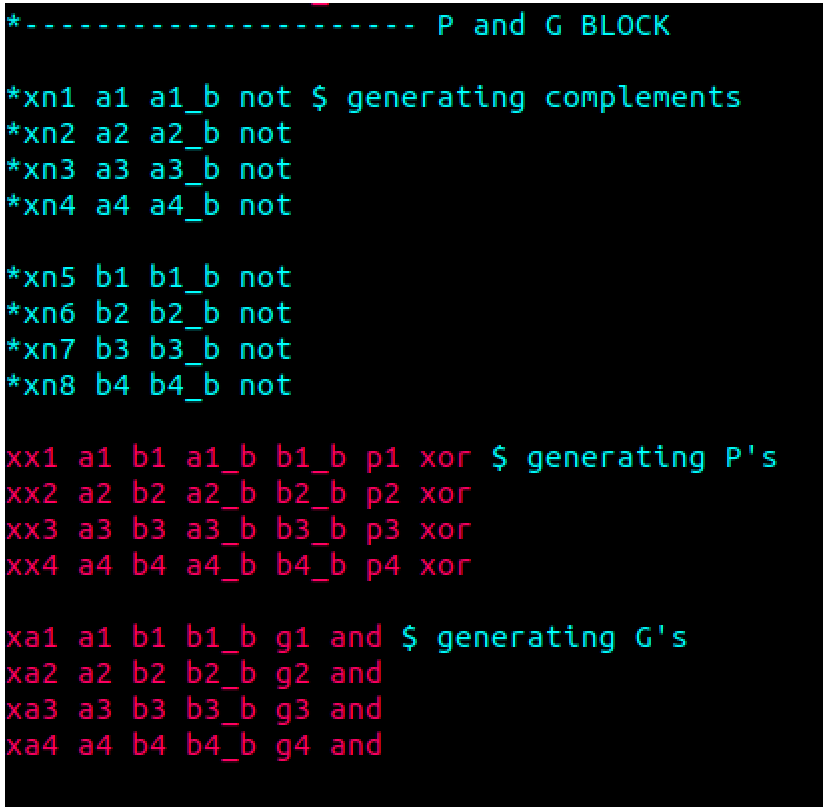
3 input NAND gate is a simple extension of 2 input NAND gate where another PMOS is connected in parallel while a NMOS is connected in series with the other set of NMOS.

PG Block



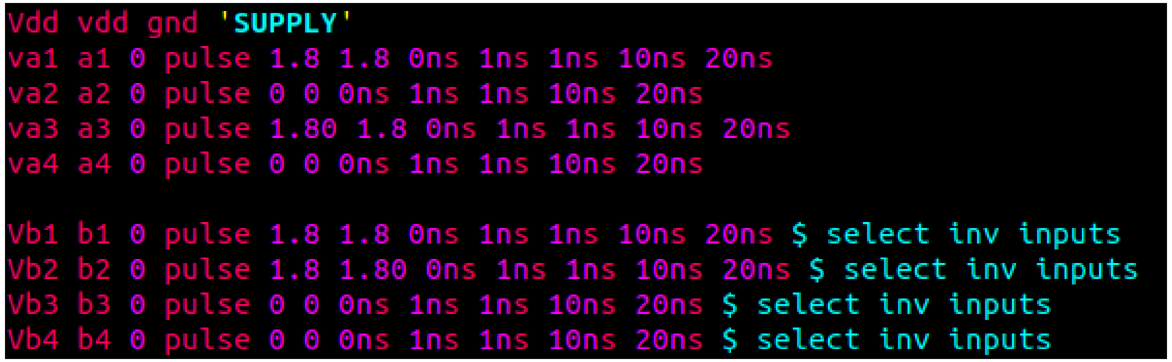
Here, i represents the ith bit.
So, 4 such blocks were used for the 4 bits giving a total of 4 XOR and 4 AND gates.

NETLIST



```
*---------------------- P and G BLOCK

*xn1 a1 a1_b not $ generating complements
*xn2 a2 a2_b not
*xn3 a3 a3_b not
*xn4 a4 a4_b not

*xn5 b1 b1_b not
*xn6 b2 b2_b not
*xn7 b3 b3_b not
*xn8 b4 b4_b not

xx1 a1 b1 a1_b b1_b p1 xor $ generating P's
xx2 a2 b2 a2_b b2_b p2 xor
xx3 a3 b3 a3_b b3_b p3 xor
xx4 a4 b4 a4_b b4_b p4 xor

xa1 a1 b1 b1_b g1 and $ generating G's
xa2 a2 b2 b2_b g2 and
xa3 a3 b3 b3_b g3 and
xa4 a4 b4 b4_b g4 and
```

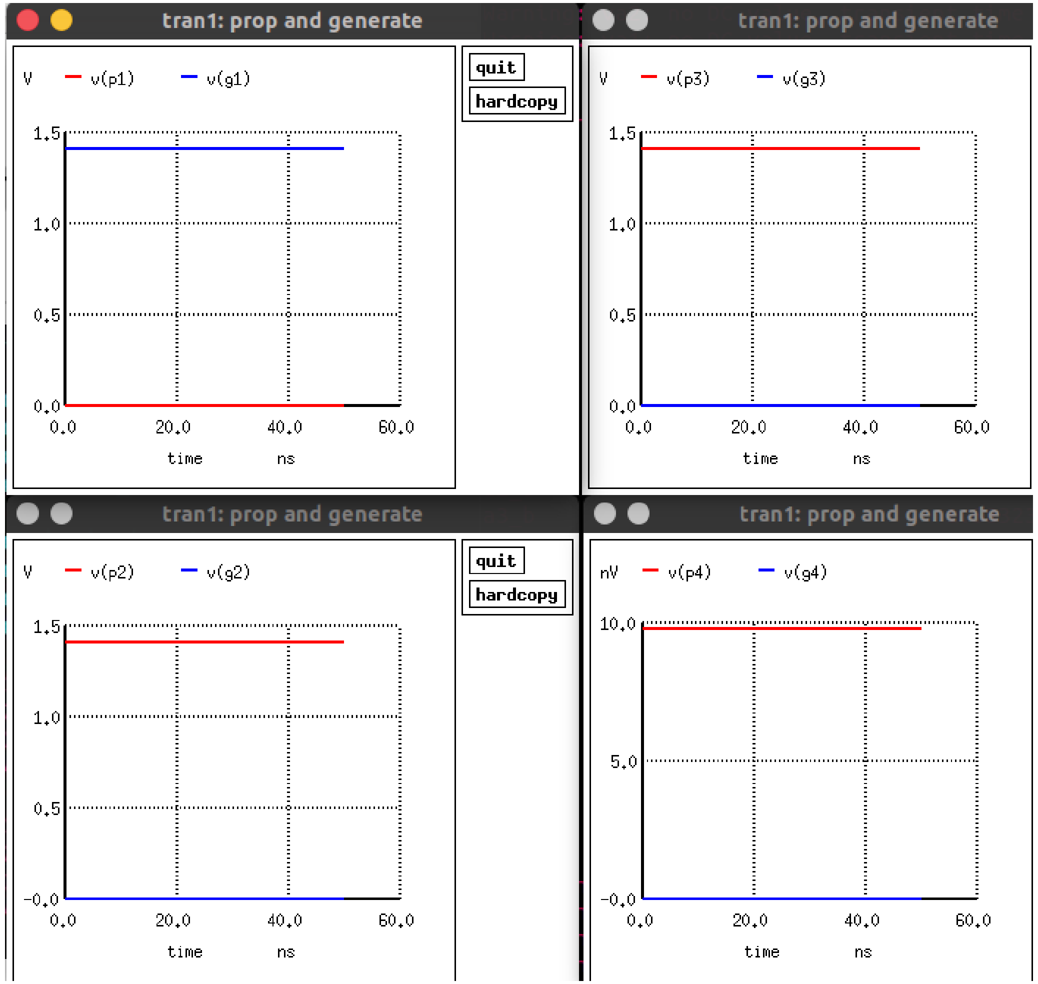
INPUTS



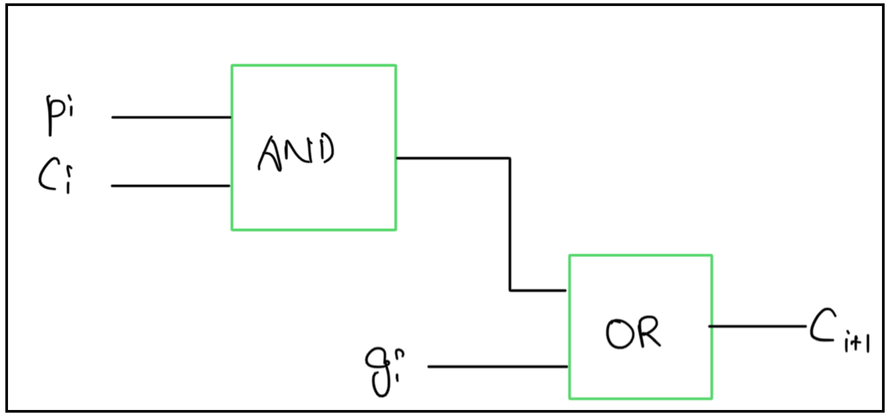
```
Vdd vdd gnd 'SUPPLY'
va1 a1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns
va2 a2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va3 a3 0 pulse 1.80 1.8 0ns 1ns 1ns 10ns 20ns
va4 a4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns

Vb1 b1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb2 b2 0 pulse 1.8 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb3 b3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb4 b4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

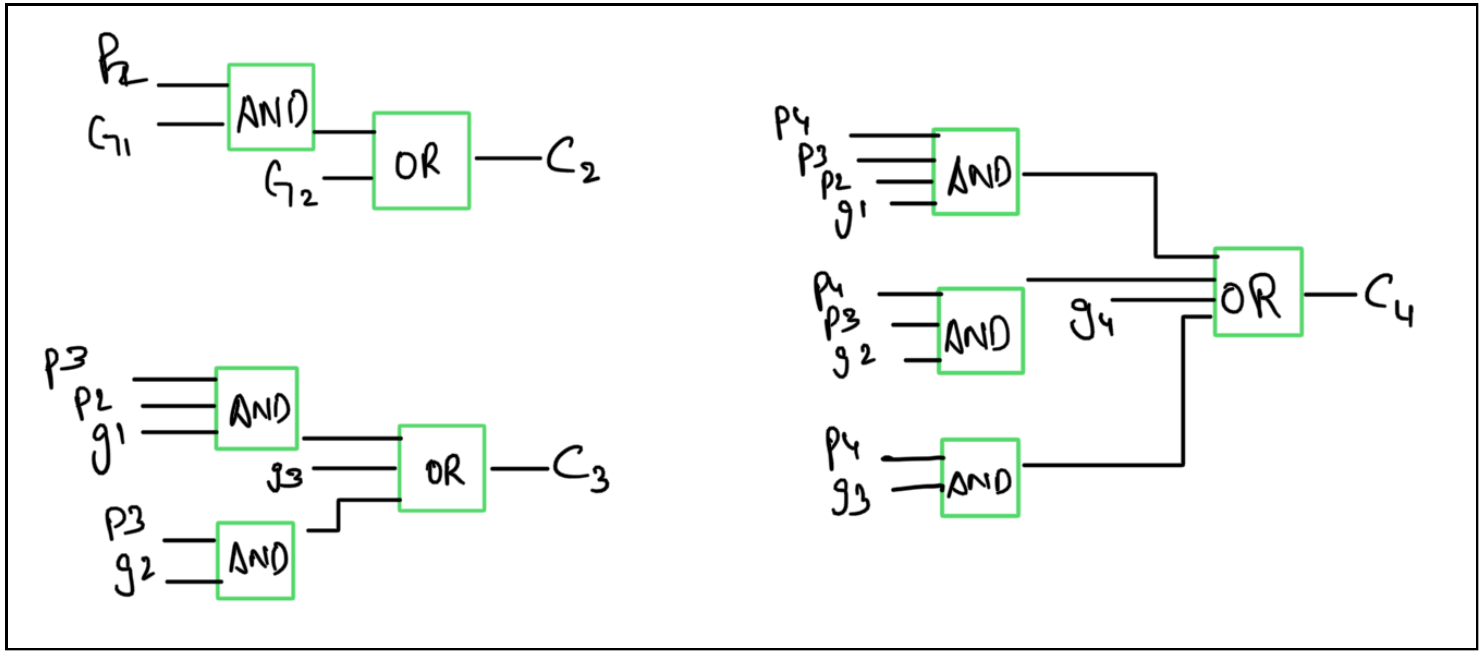
*All 4 possible combinations possible are given

Plots

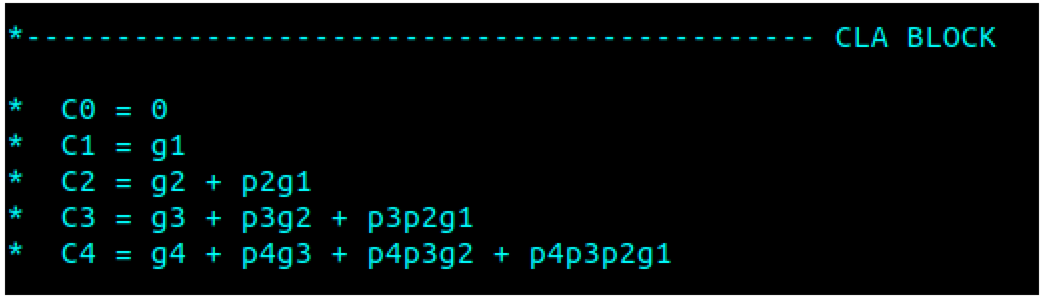

tran1: prop and generate
V
v(p1)
v(g1)
1.5
1.0
0.5
0.0
0.0
20.0
40.0
60.0
time
ns
quit
hardcopy
tran1: prop and generate
V
v(p3)
v(g3)
1.5
1.0
0.5
0.0
0.0
20.0
40.0
60.0
time
ns
tran1: prop and generate
V
v(p2)
v(g2)
1.5
1.0
0.5
-0.0
0.0
20.0
40.0
60.0
time
ns
quit
hardcopy
tran1: prop and generate
nV
v(p4)
v(g4)
10.0
5.0
-0.0
0.0
20.0
40.0
60.0
time
ns

CLA Block



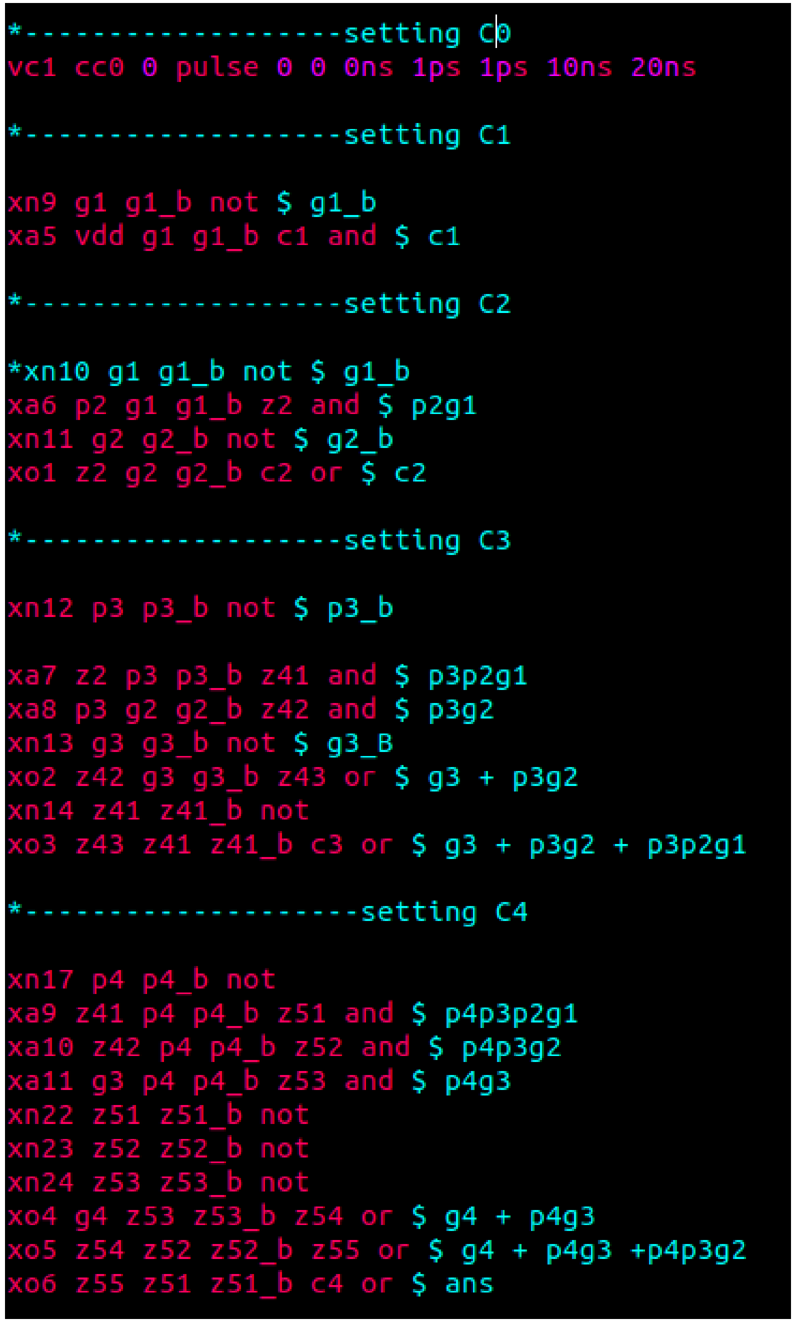


Here, we use the sample block above for generating c1 and further carries. Here, 3,4 input gates are again implemented using their primitive 2 input gates. Also I've considered the input carry C0 = 0 for all cases.



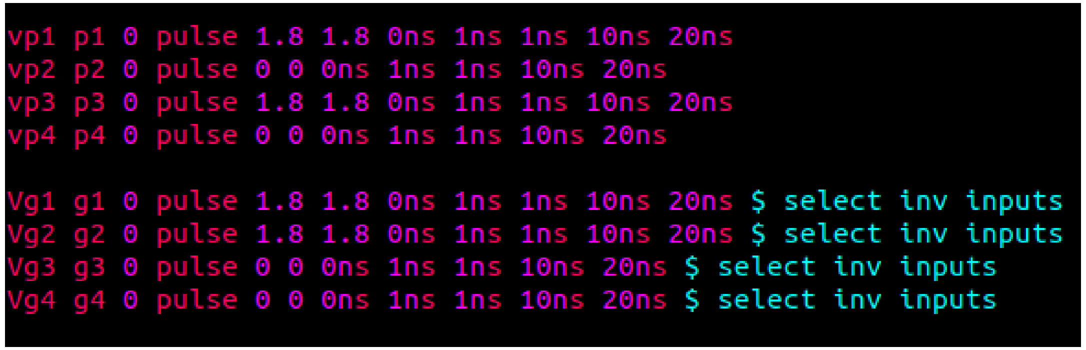
```
*------------------------------------------ CLA BLOCK

*  C0 = 0
*  C1 = g1
*  C2 = g2 + p2g1
*  C3 = g3 + p3g2 + p3p2g1
*  C4 = g4 + p4g3 + p4p3g2 + p4p3p2g1
```

NETLIST



```
*-------------------setting C0
vc1 cc0 0 pulse 0 0 0ns 1ps 1ps 10ns 20ns

*-------------------setting C1

xn9 g1 g1_b not $ g1_b
xa5 vdd g1 g1_b c1 and $ c1

*-------------------setting C2

*xn10 g1 g1_b not $ g1_b
xa6 p2 g1 g1_b z2 and $ p2g1
xn11 g2 g2_b not $ g2_b
xo1 z2 g2 g2_b c2 or $ c2

*-------------------setting C3

xn12 p3 p3_b not $ p3_b

xa7 z2 p3 p3_b z41 and $ p3p2g1
xa8 p3 g2 g2_b z42 and $ p3g2
xn13 g3 g3_b not $ g3_B
xo2 z42 g3 g3_b z43 or $ g3 + p3g2
xn14 z41 z41_b not
xo3 z43 z41 z41_b c3 or $ g3 + p3g2 + p3p2g1

*-------------------setting C4

xn17 p4 p4_b not
xa9 z41 p4 p4_b z51 and $ p4p3p2g1
xa10 z42 p4 p4_b z52 and $ p4p3g2
xa11 g3 p4 p4_b z53 and $ p4g3
xn22 z51 z51_b not
xn23 z52 z52_b not
xn24 z53 z53_b not
xo4 g4 z53 z53_b z54 or $ g4 + p4g3
xo5 z54 z52 z52_b z55 or $ g4 + p4g3 +p4p3g2
xo6 z55 z51 z51_b c4 or $ ans
```

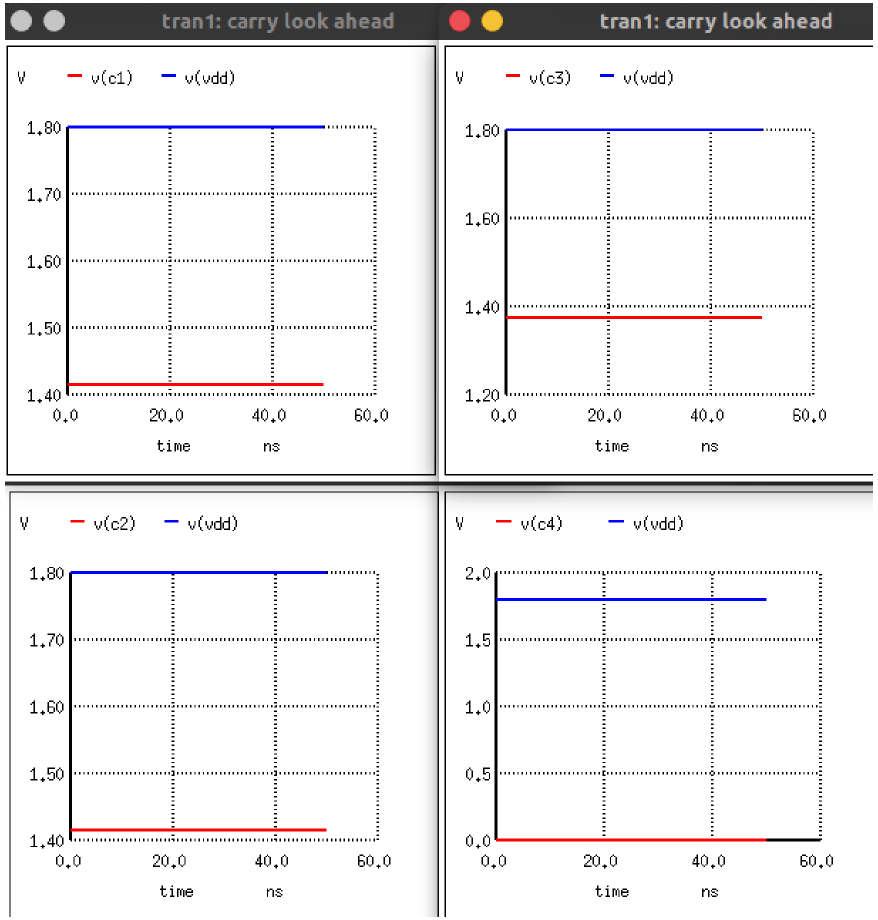
INPUTS



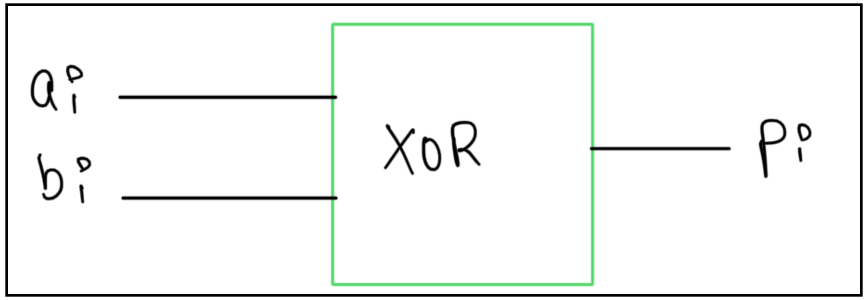
```
vp1 p1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns
vp2 p2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
vp3 p3 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns
vp4 p4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns

Vg1 g1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg2 g2 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg3 g3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg4 g4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

*All 4 possible combinations possible are given

Plots

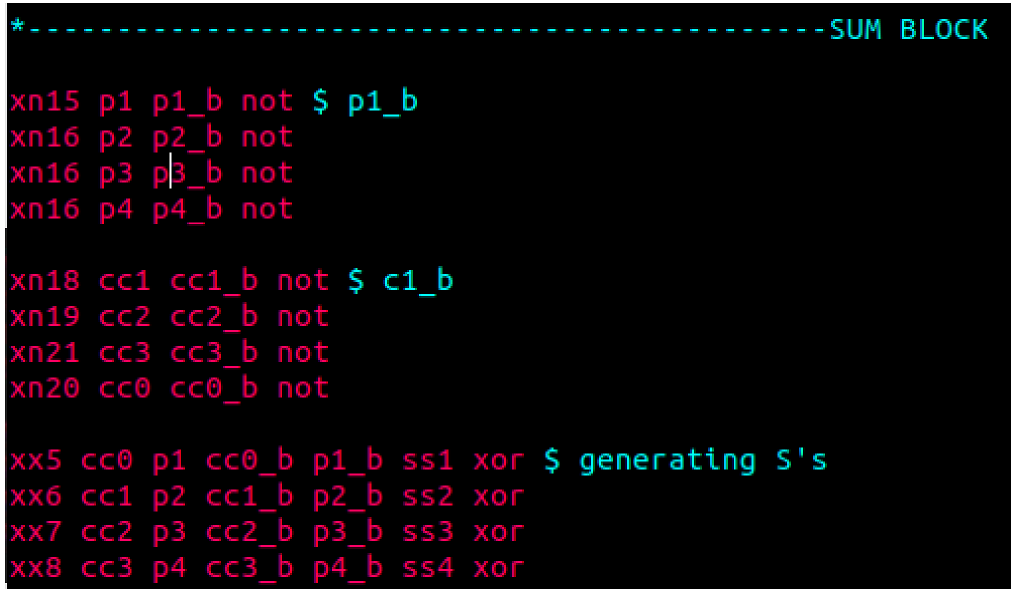

tran1: carry look ahead
tran1: carry look ahead
V
v(c1)
v(vdd)
1.80
1.70
1.60
1.50
1.40
0.0
20.0
40.0
60.0
time
ns
V
v(c3)
v(vdd)
1.80
1.60
1.40
1.20
0.0
20.0
40.0
60.0
time
ns
V
v(c2)
v(vdd)
1.80
1.70
1.60
1.50
1.40
0.0
20.0
40.0
60.0
time
ns
V
v(c4)
v(vdd)
2.0
1.5
1.0
0.5
0.0
0.0
20.0
40.0
60.0
time
ns

SUM Block + BUFFer



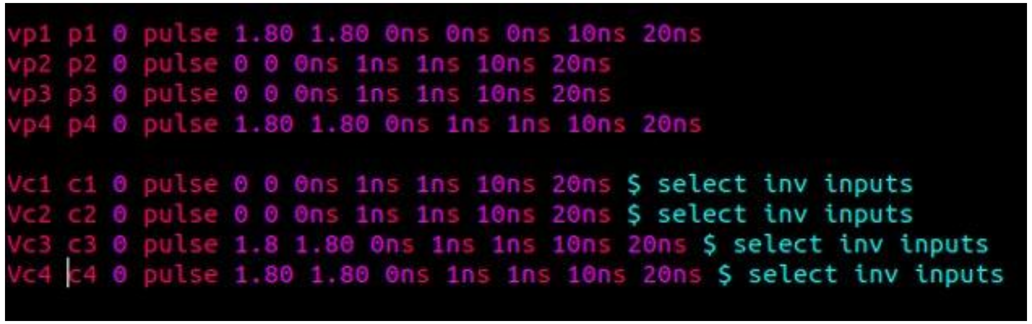
Here, i represents the ith bit.
So, 4 such blocks were used for the 4 bits requiring a total of 4 XOR gates.

NETLIST



```
*--------------------------------------------SUM BLOCK

xn15 p1 p1_b not $ p1_b
xn16 p2 p2_b not
xn16 p3 p3_b not
xn16 p4 p4_b not

xn18 cc1 cc1_b not $ c1_b
xn19 cc2 cc2_b not
xn21 cc3 cc3_b not
xn20 cc0 cc0_b not

xx5 cc0 p1 cc0_b p1_b ss1 xor $ generating S's
xx6 cc1 p2 cc1_b p2_b ss2 xor
xx7 cc2 p3 cc2_b p3_b ss3 xor
xx8 cc3 p4 cc3_b p4_b ss4 xor
```

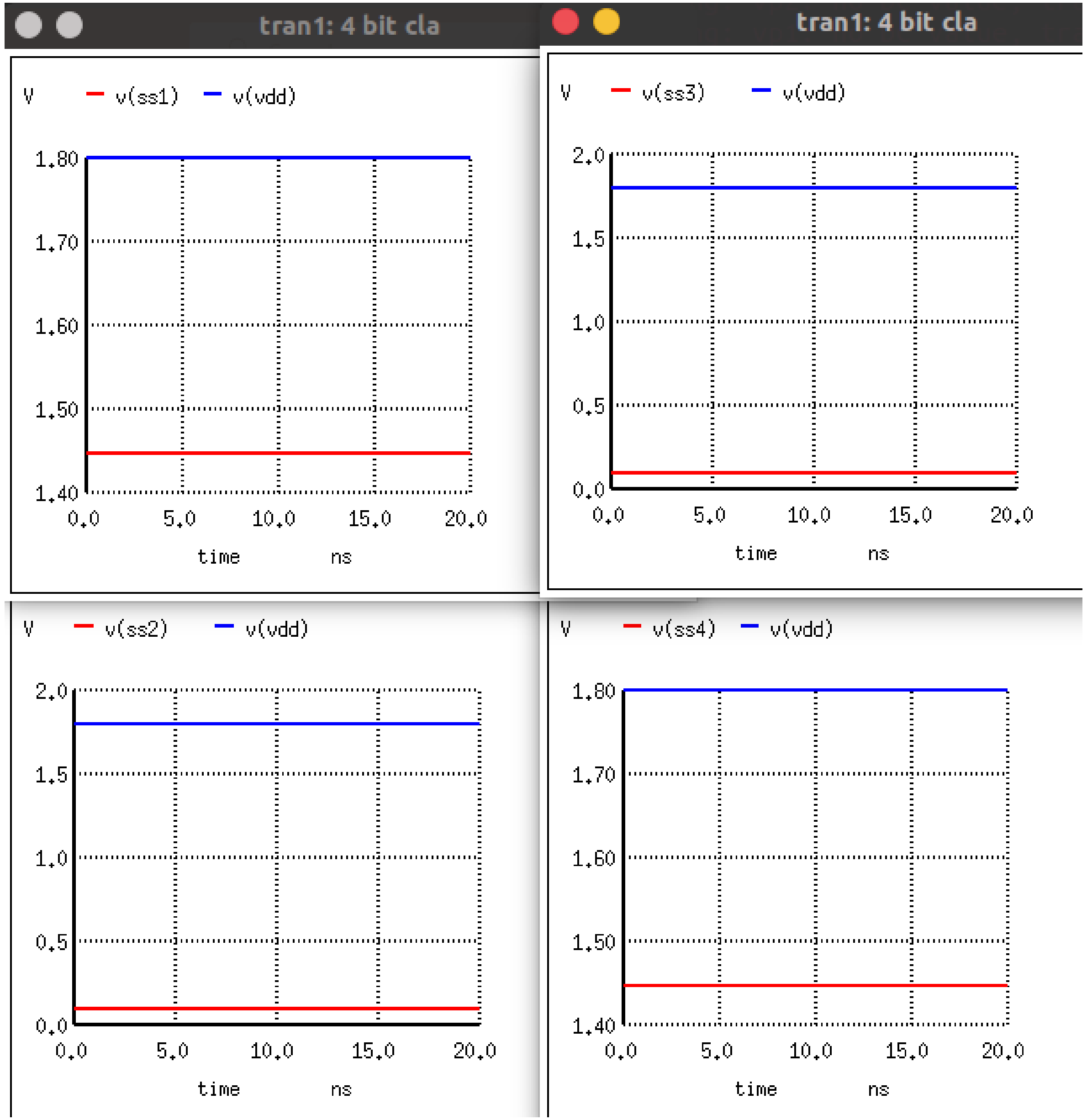
INPUTS



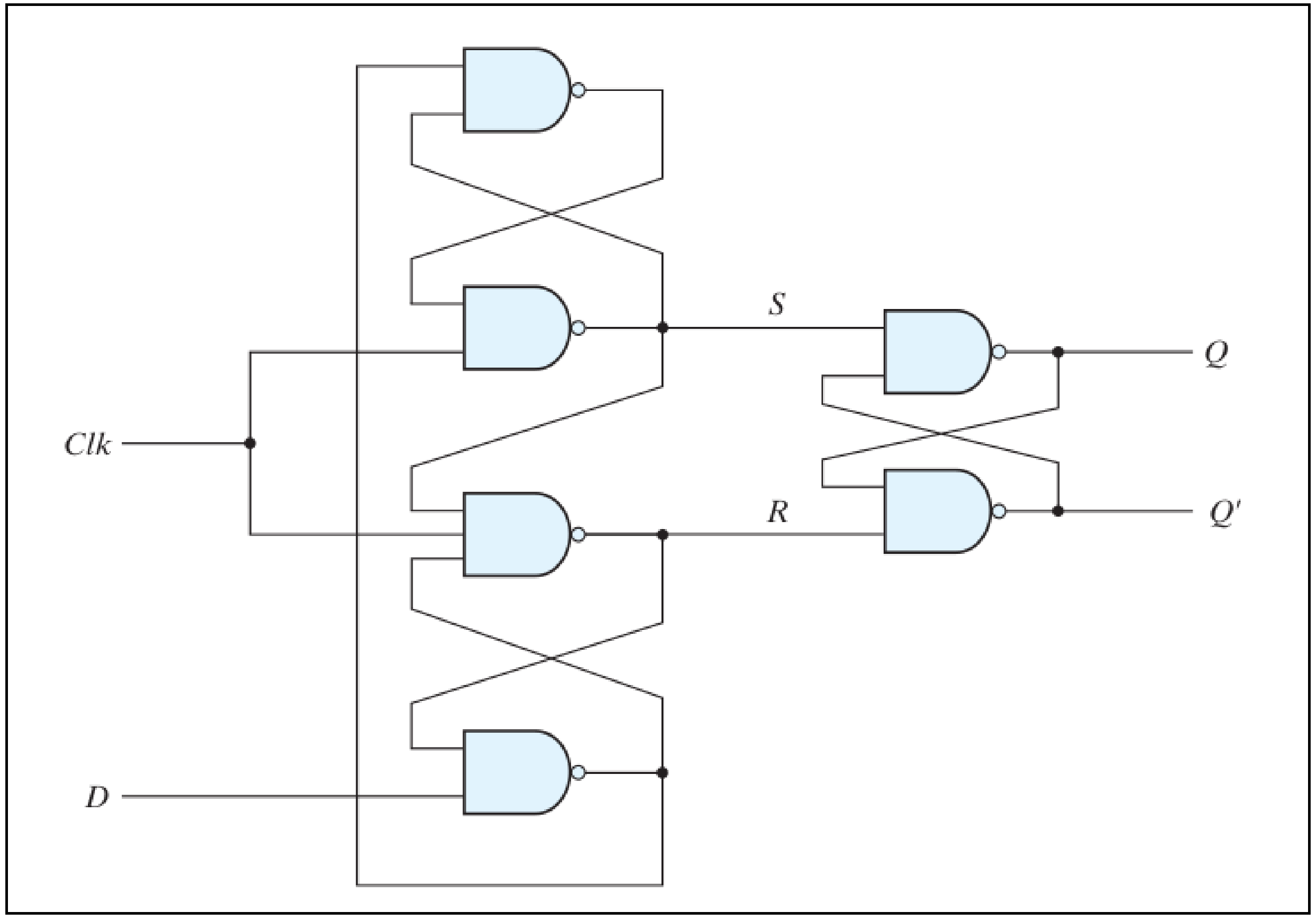
```
vp1 p1 0 pulse 1.80 1.80 0ns 0ns 0ns 10ns 20ns
vp2 p2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
vp3 p3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
vp4 p4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns

Vc1 c1 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vc2 c2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vc3 c3 0 pulse 1.8 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vc4 c4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

*All 4 possible combinations possible are given

Plots


tran1: 4 bit cla
tran1: 4 bit cla
V
v(ss1)
v(vdd)
1.80
1.70
1.60
1.50
1.40
0.0
5.0
10.0
15.0
20.0
time
ns
V
v(ss3)
v(vdd)
2.0
1.5
1.0
0.5
0.0
0.0
5.0
10.0
15.0
20.0
time
ns
V
v(ss2)
v(vdd)
2.0
1.5
1.0
0.5
0.0
0.0
5.0
10.0
15.0
20.0
time
ns
V
v(ss4)
v(vdd)
1.80
1.70
1.60
1.50
1.40
0.0
5.0
10.0
15.0
20.0
time
ns

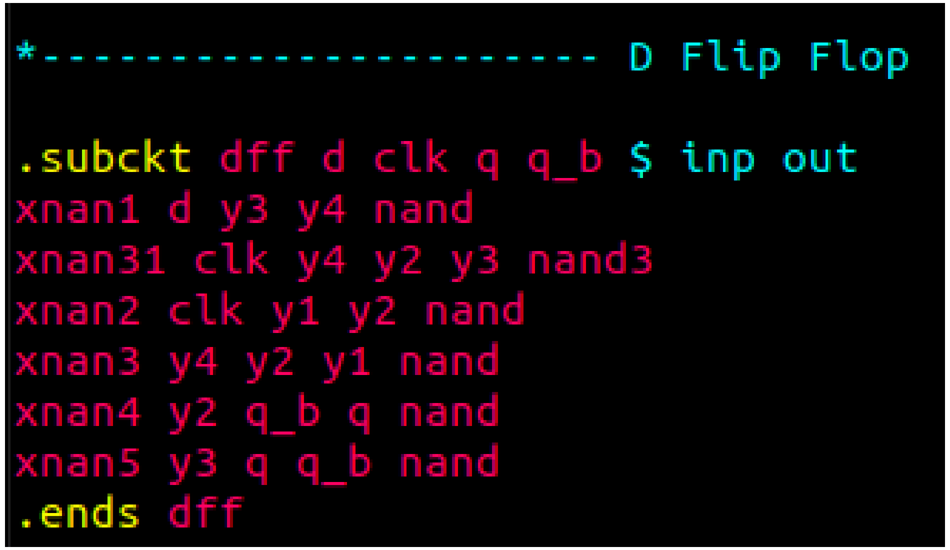
DFF Block



*Ref: Morris Mano, Digital Systems

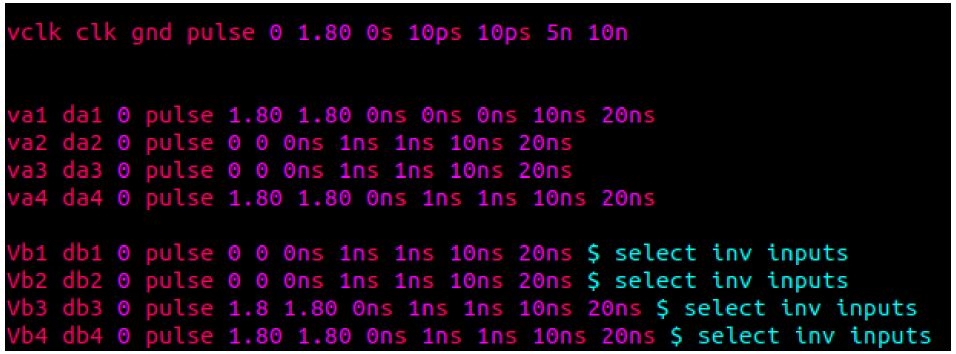
Here , D is our input to the block while Q is the required input. Total of 5 2-input NAND gates and a 3 input NAND gate are used.

NETLIST



```
*---------------------- D Flip Flop

.subckt dff d clk q q_b $ inp out
xnan1 d y3 y4 nand
xnan31 clk y4 y2 y3 nand3
xnan2 clk y1 y2 nand
xnan3 y4 y2 y1 nand
xnan4 y2 q_b q nand
xnan5 y3 q q_b nand
.ends dff
```

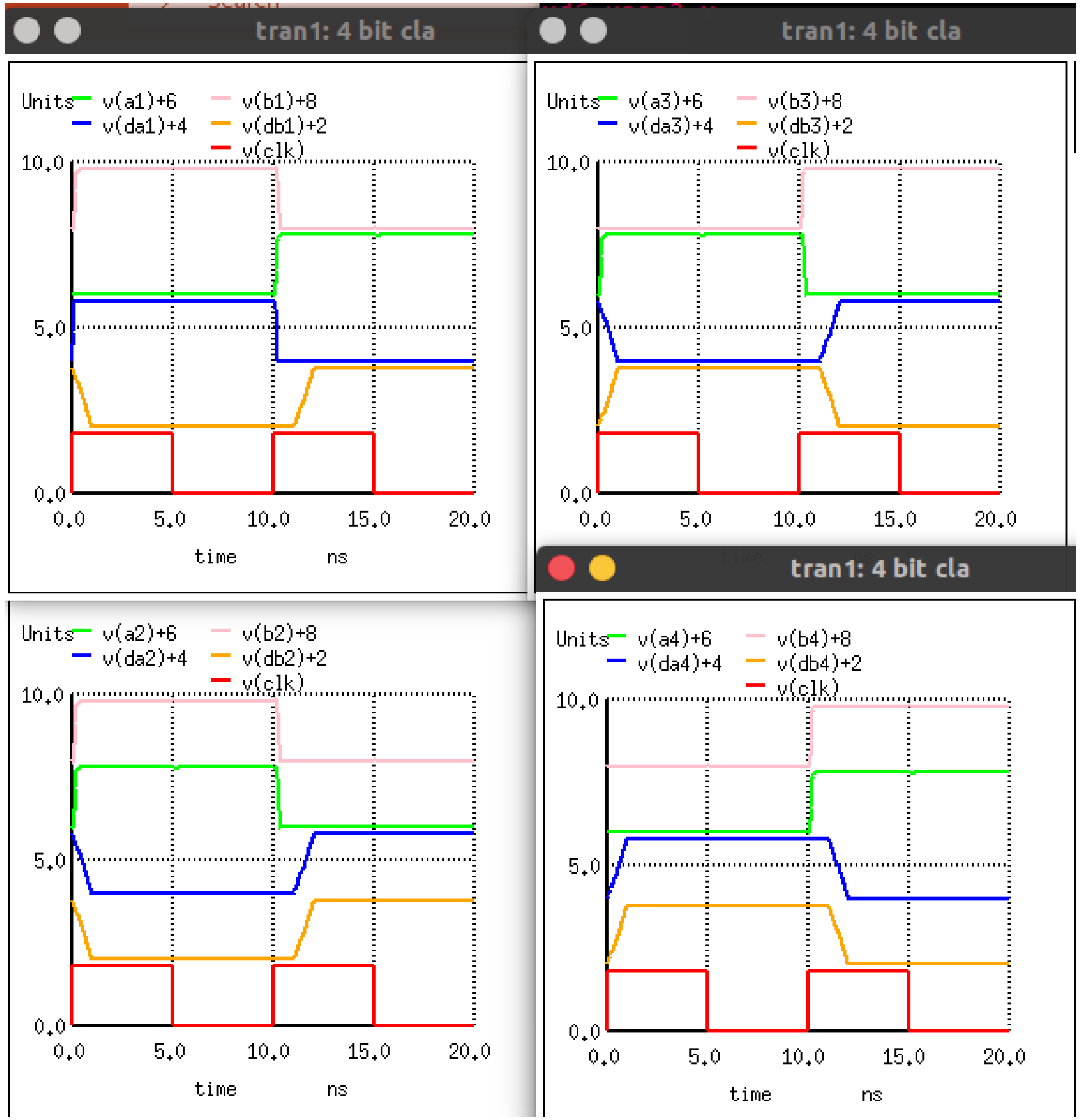
INPUTS



```
vclk clk gnd pulse 0 1.80 0s 10ps 10ps 5n 10n


va1 da1 0 pulse 1.80 1.80 0ns 0ns 0ns 10ns 20ns
va2 da2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va3 da3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va4 da4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns

Vb1 db1 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb2 db2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb3 db3 0 pulse 1.8 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb4 db4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

*All 4 possible combinations possible are given

Plots


tran1: 4 bit cla
Units v(a1)+6 v(b1)+8
v(da1)+4 v(db1)+2
v(clk)
10.0
5.0
0.0
0.0 5.0 10.0 15.0 20.0
time ns
tran1: 4 bit cla
Units v(a3)+6 v(b3)+8
v(da3)+4 v(db3)+2
v(clk)
10.0
5.0
0.0
0.0 5.0 10.0 15.0 20.0
Units v(a2)+6 v(b2)+8
v(da2)+4 v(db2)+2
v(clk)
10.0
5.0
0.0
0.0 5.0 10.0 15.0 20.0
time ns
tran1: 4 bit cla
Units v(a4)+6 v(b4)+8
v(da4)+4 v(db4)+2
v(clk)
10.0
5.0
0.0
0.0 5.0 10.0 15.0 20.0
time ns

Question 4

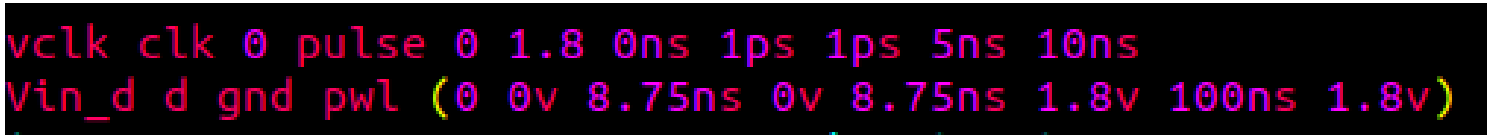
For D-flip-flop, find its setup time, hold time and clock to Q delay from NGSPICE simulations.

Setup Time: the amount of time the data at the input (D) that must be stable before the active edge of clock

Hold Time: the amount of time the data at the input (D) that must be stable after the active edge of clock.

Clk to Q delay: It’s the propogation delay wrt clock positive edge for the input (D) to appear at the output (Q) of the flipflop

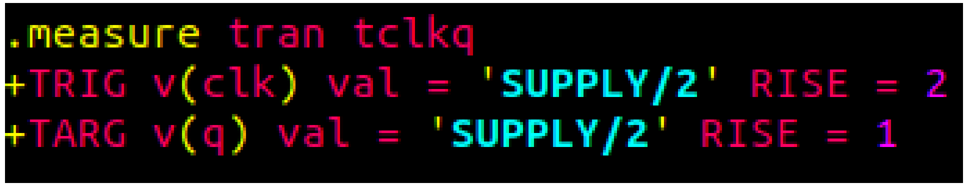
Clk to Q delay



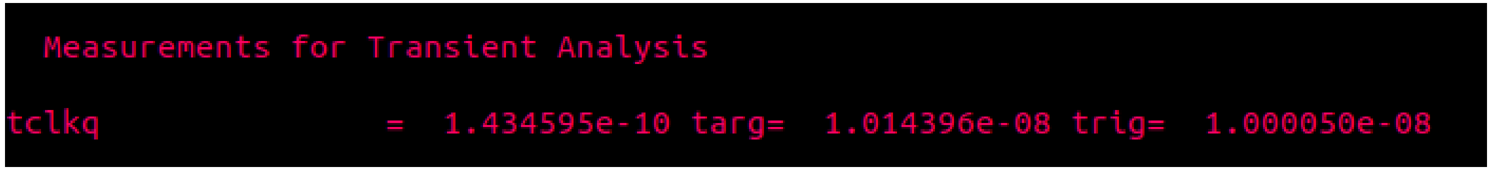
```
vclk clk 0 pulse 0 1.8 0ns 1ps 1ps 5ns 10ns
Vin_d d gnd pwl (0 0v 8.75ns 0v 8.75ns 1.8v 100ns 1.8v)
```

Clk and Input Parameters

Measurement Criteria



```
.measure tran tclkq
+TRIG v(clk) val = 'SUPPLY/2' RISE = 2
+TARG v(q) val = 'SUPPLY/2' RISE = 1
```

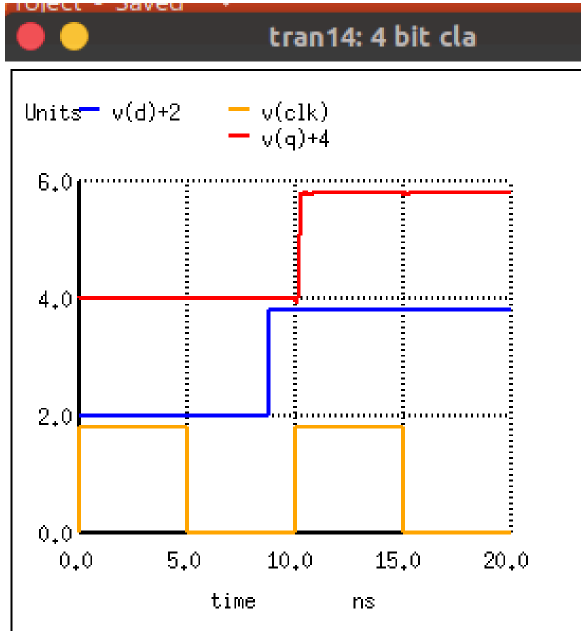


```
 Measurements for Transient Analysis

tclkq                =  1.434595e-10 targ=  1.014396e-08 trig=  1.000050e-08
```

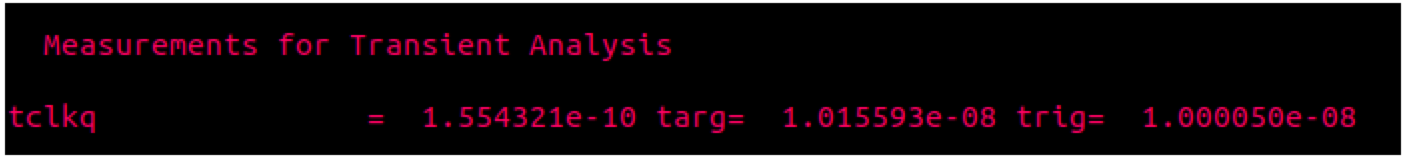
Measured Values

Obtained value of tclkq = 0.143ns.
Here we have our 2nd rising edge of clk at 10 ns.It can be noted that input is given after a delay of 8.75 which is away from the setup time that is found next

Setup Time



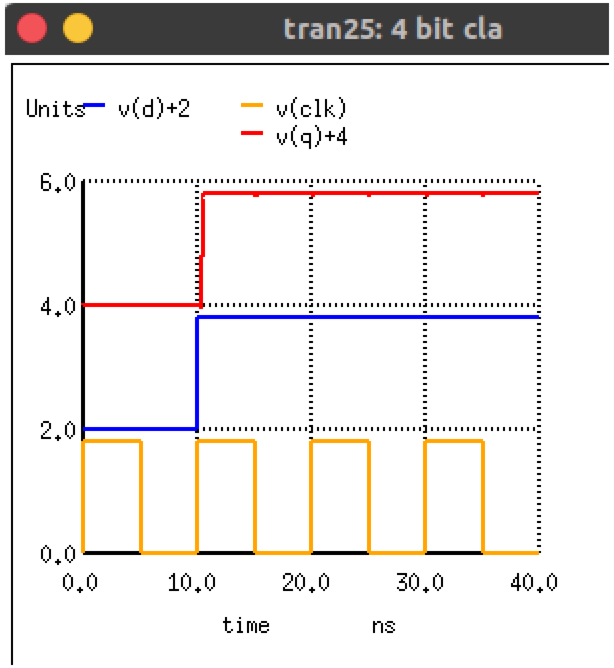
Measured Values



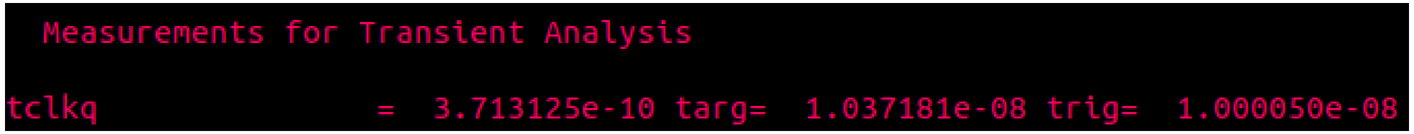
```
  Measurements for Transient Analysis

tclkq               =  1.554321e-10 targ=  1.015593e-08 trig=  1.000050e-08
```

Here input at D is applied after a delay of 9.86ns for which we observe tclkq = 0.155ns which is a rise of 0.012 ns ie. ~10% change of original value. Therefore Setup time = clk time – delay time I.e 10 – 9.86 = 0.14ns

Hold Time



Measured Values

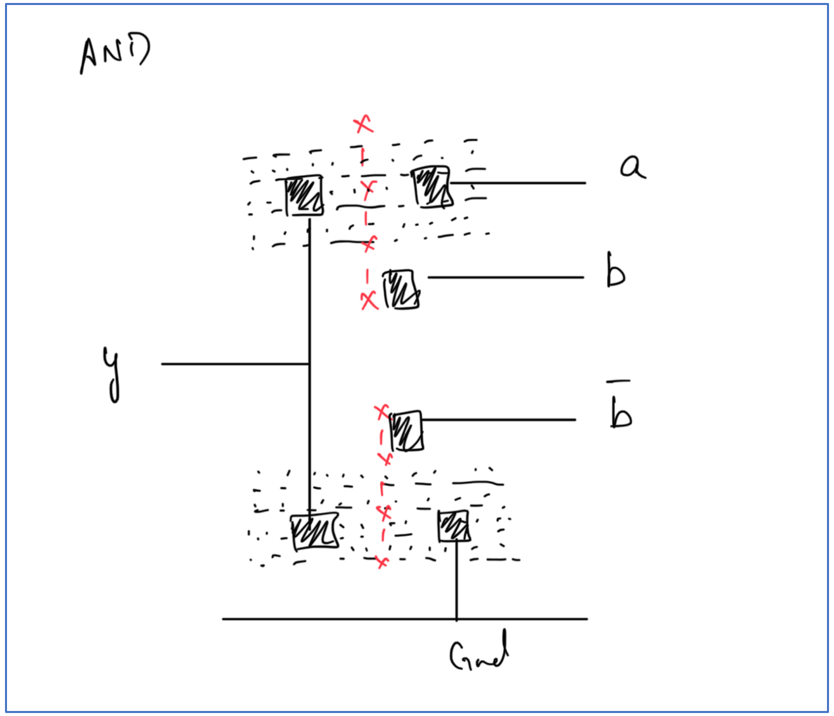


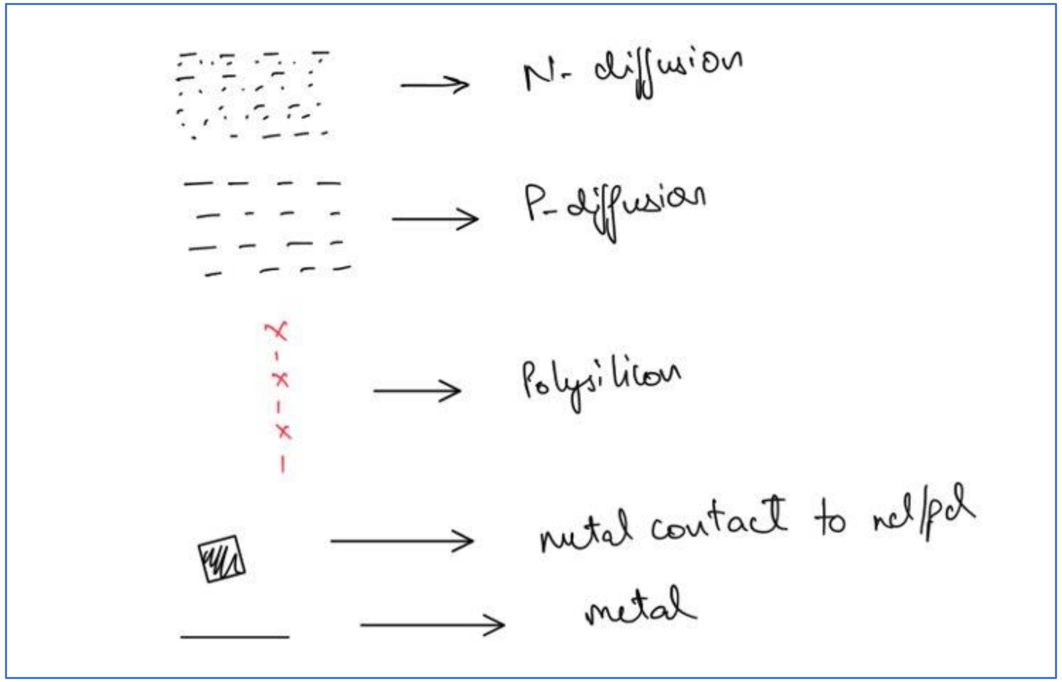
```
 Measurements for Transient Analysis

tclkq                =  3.713125e-10 targ=  1.037181e-08 trig=  1.000050e-08
```

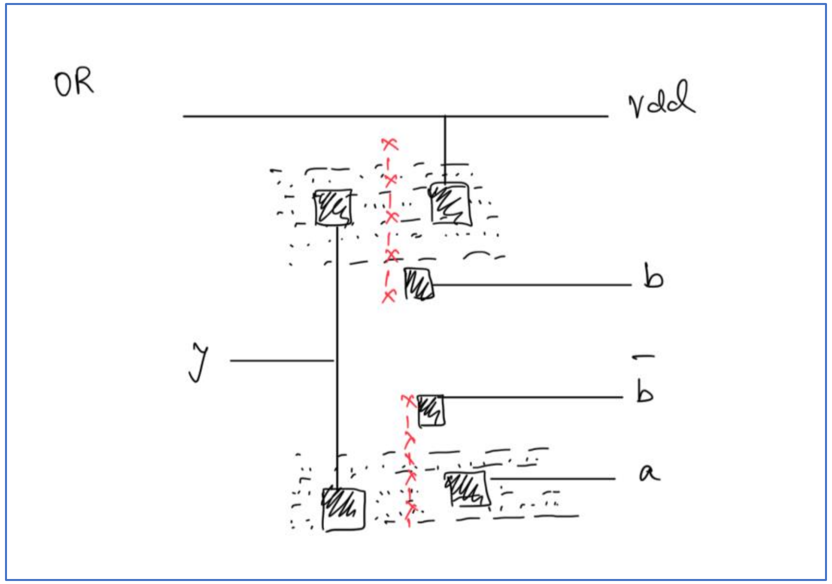
Here input at D is applied after a delay of 10.0175ns for which we observe the output to change in the same clock cycle with a delay of 0.371ns. Therefore, Hold time = 0.0175ns

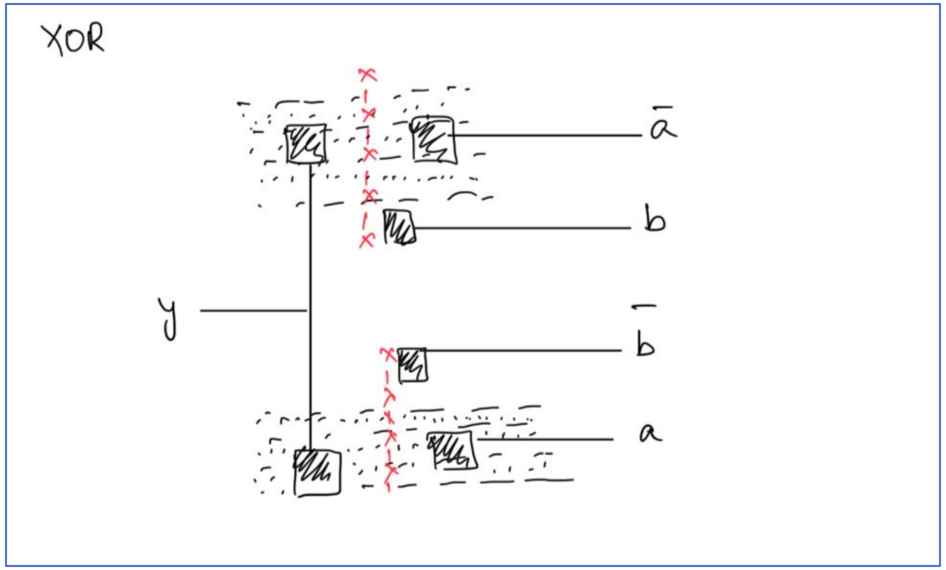
Question 5

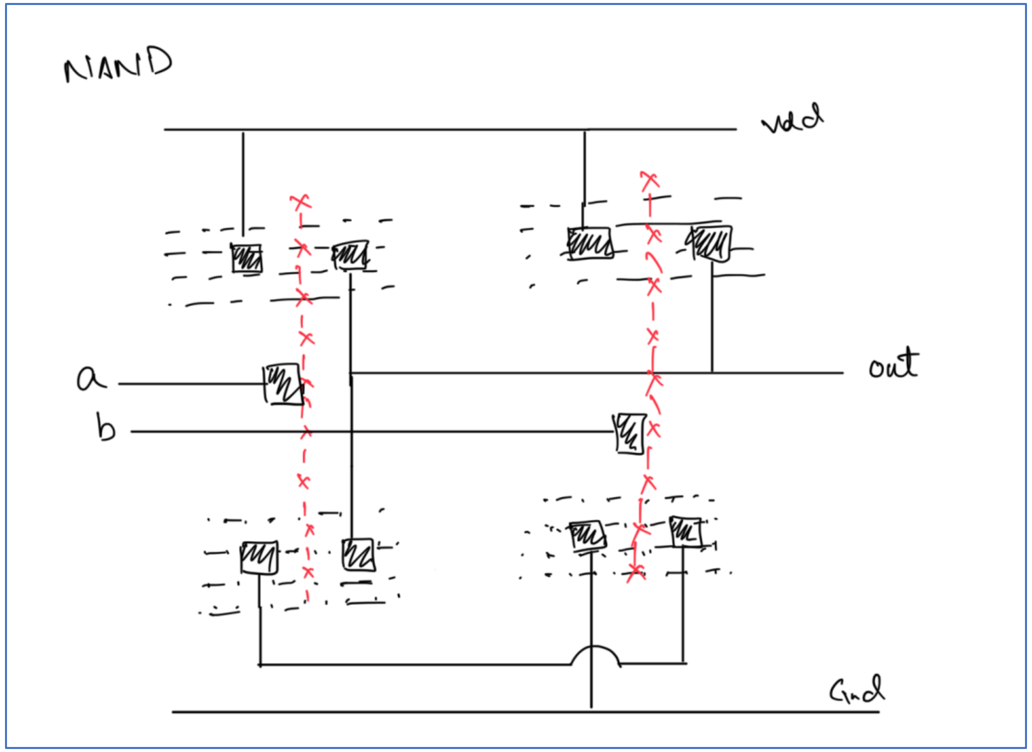
- Give stick diagrams of all unique gates in your design.

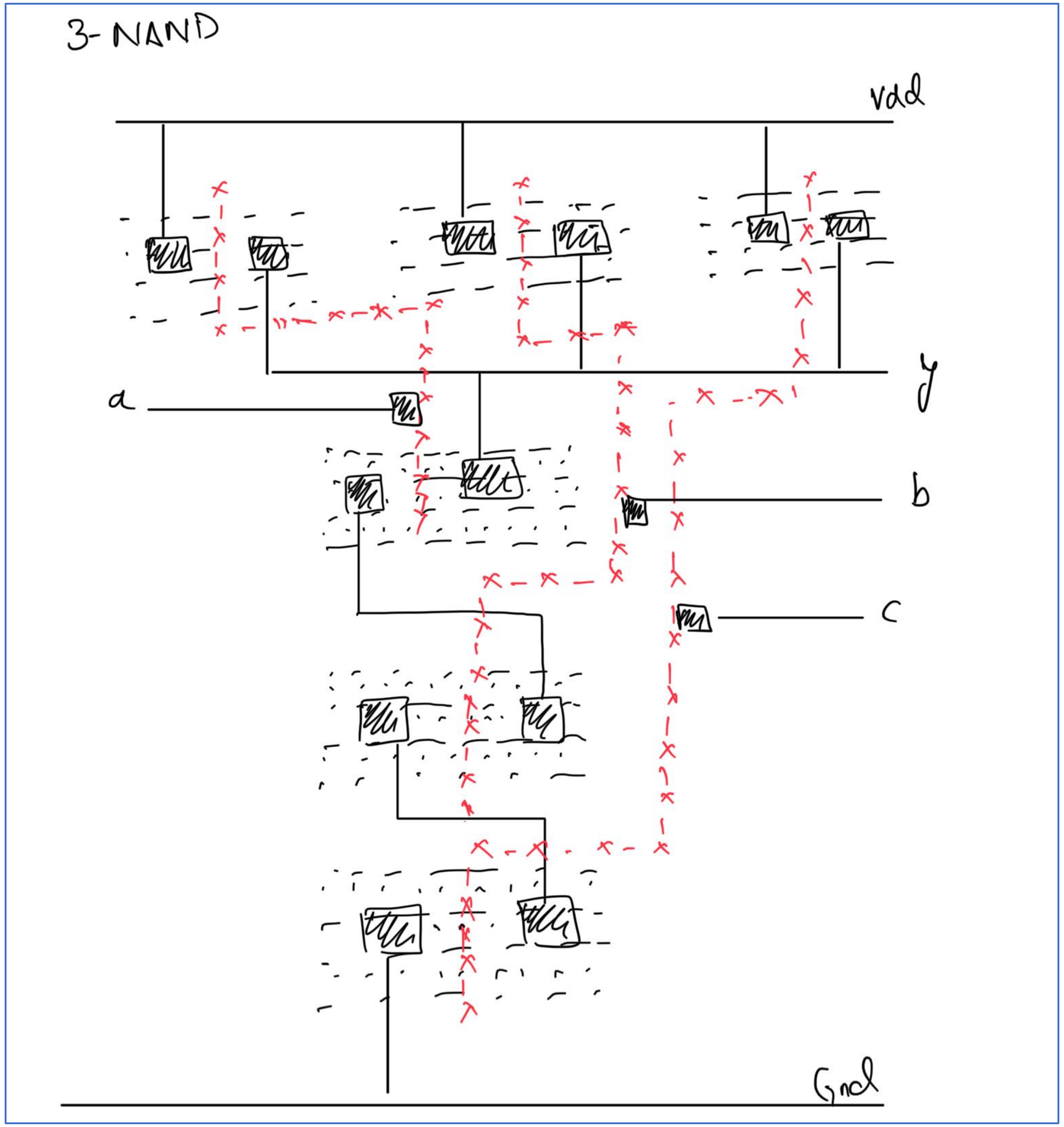

AND
a
b
y
b̄
Gnd


N- diffusion
P- diffusion
Polysilicon
metal contact to nd/pd
metal


OR
vdd
b
y
$\bar{b}$
a


XOR
$\bar{a}$
b
y
$\bar{b}$
a


NAND
vdd
a
b
out
Gnd

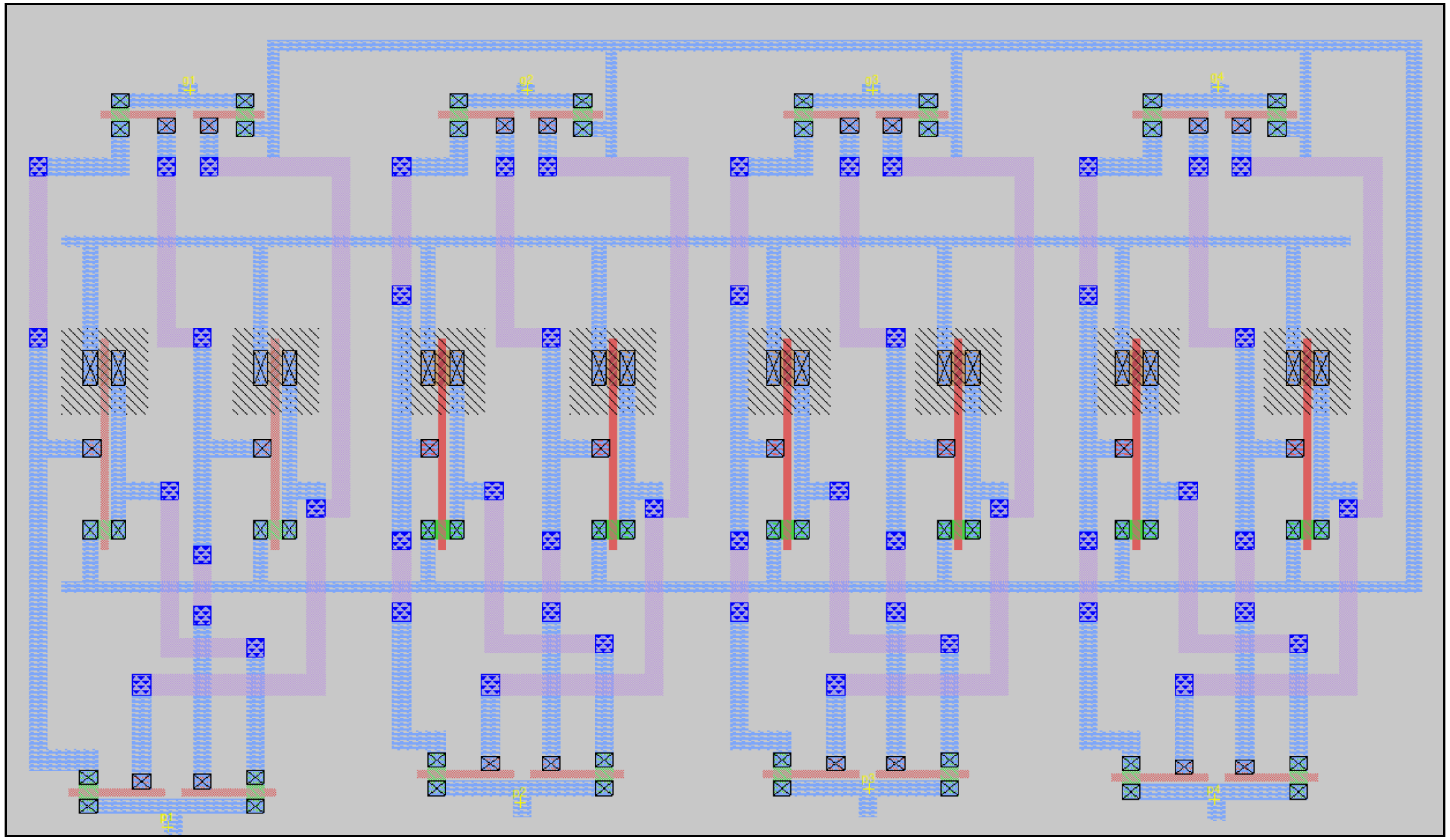

3-NAND
vdd
a
y
b
c
Gnd

Question 6

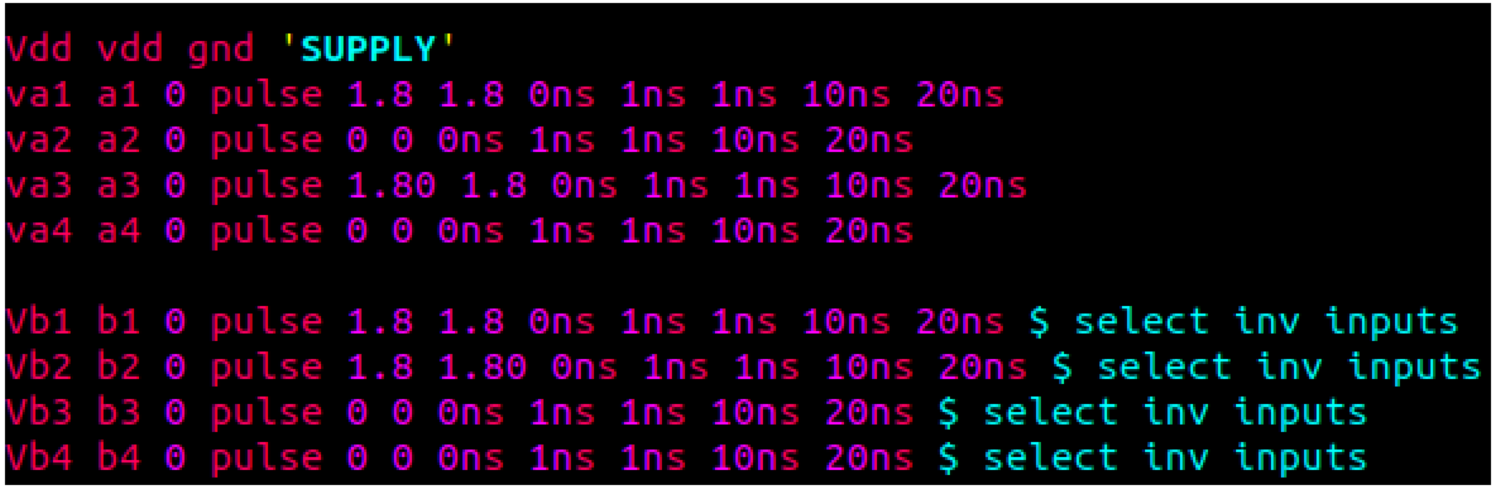
Layout each block using MAGIC layout editor and previously given technology file. Perform post layout extraction and compare the results with schematic simulation.

Technology file: *TSMC_180nm.txt*

Magic Layout for PG Block



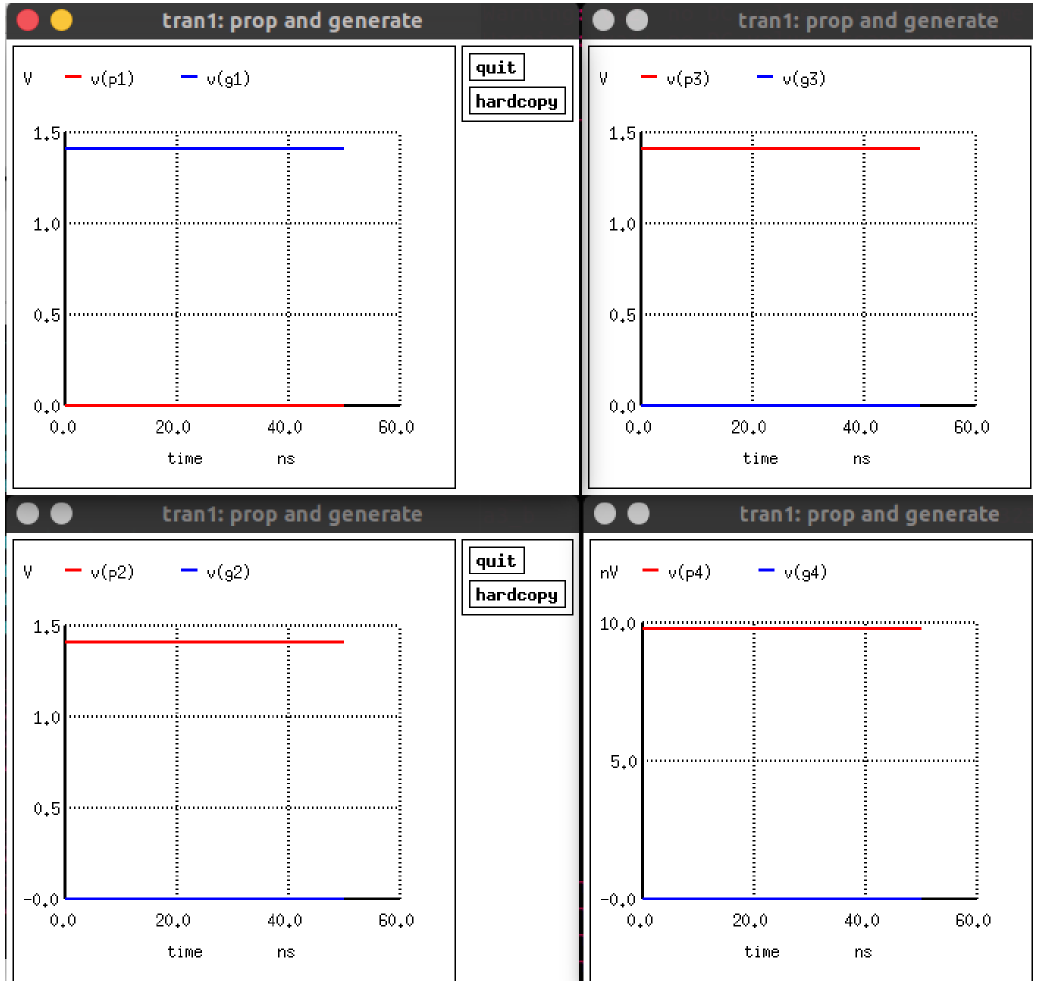
Inputs:



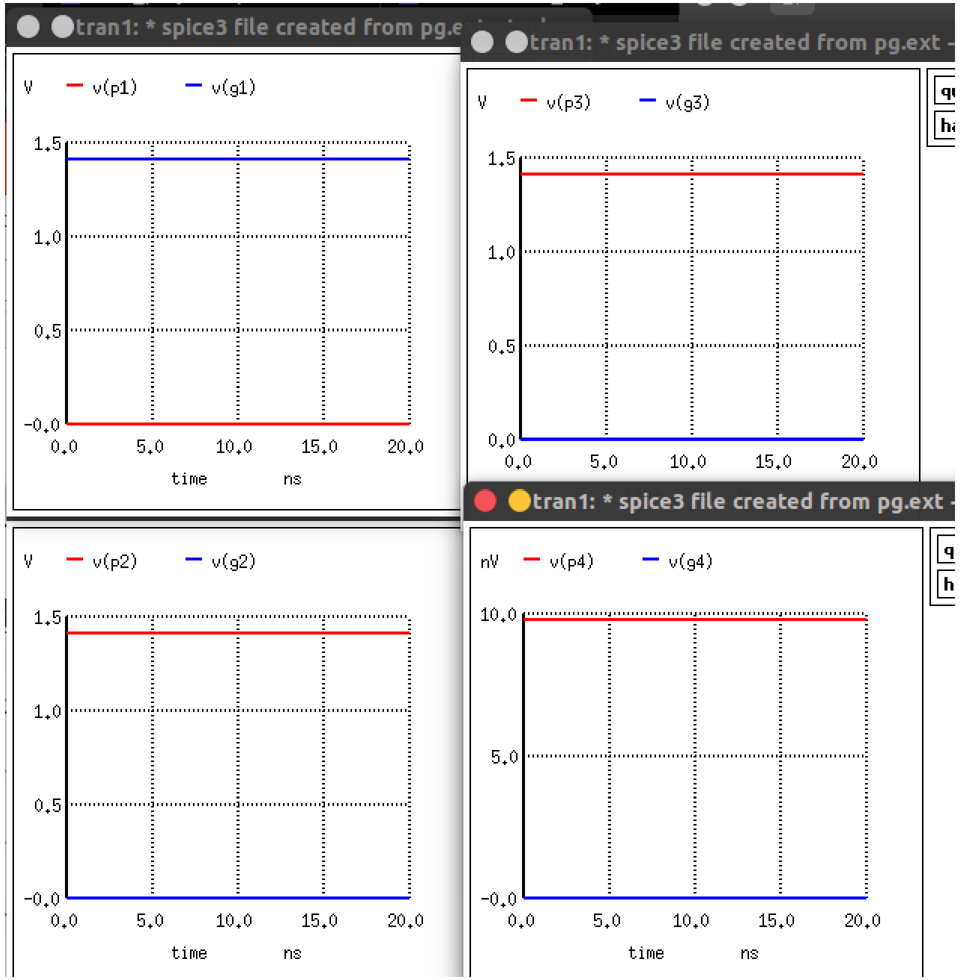
```
Vdd vdd gnd 'SUPPLY'
va1 a1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns
va2 a2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va3 a3 0 pulse 1.80 1.8 0ns 1ns 1ns 10ns 20ns
va4 a4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns

Vb1 b1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb2 b2 0 pulse 1.8 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb3 b3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb4 b4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

Plots

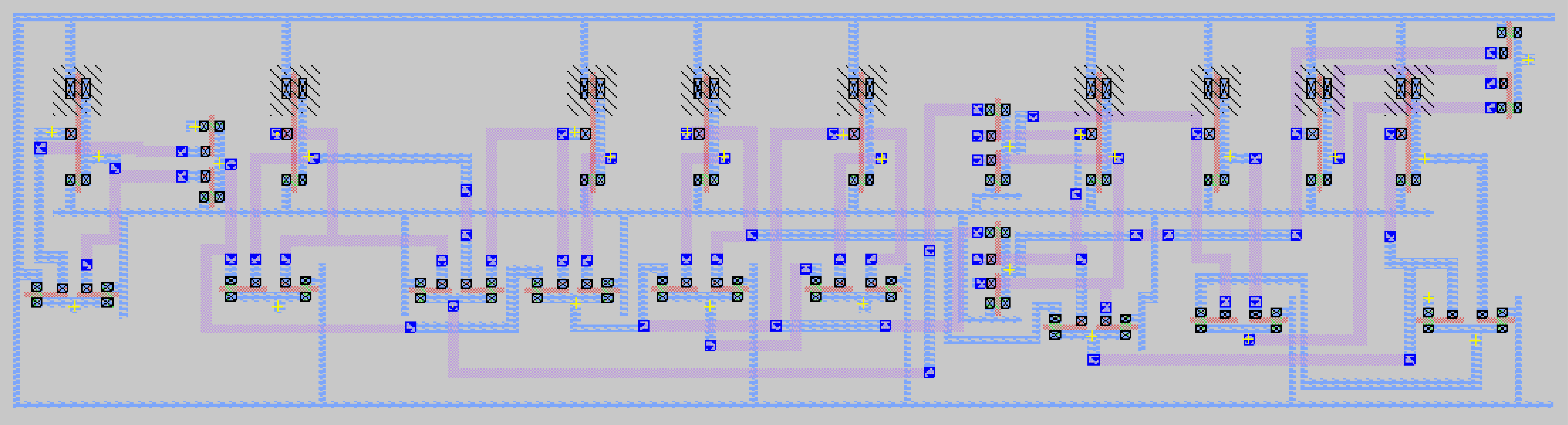
Pre-Layout



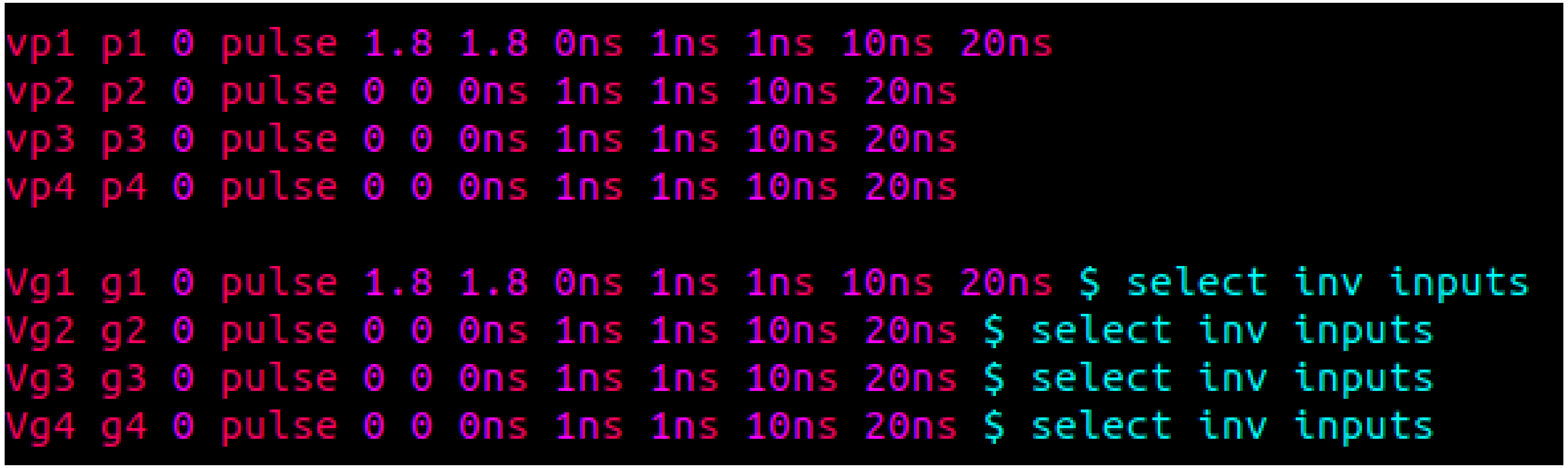
Post-Layout


tran1: * spice3 file created from pg.e
V
v(p1)
v(g1)
1.5
1.0
0.5
-0.0
0.0
5.0
10.0
15.0
20.0
time
ns
tran1: * spice3 file created from pg.ext
V
v(p3)
v(g3)
V
v(p2)
v(g2)
tran1: * spice3 file created from pg.ext
nV
v(p4)
v(g4)
10.0
5.0
-0.0

Magic Layout for CLA Block



Inputs:

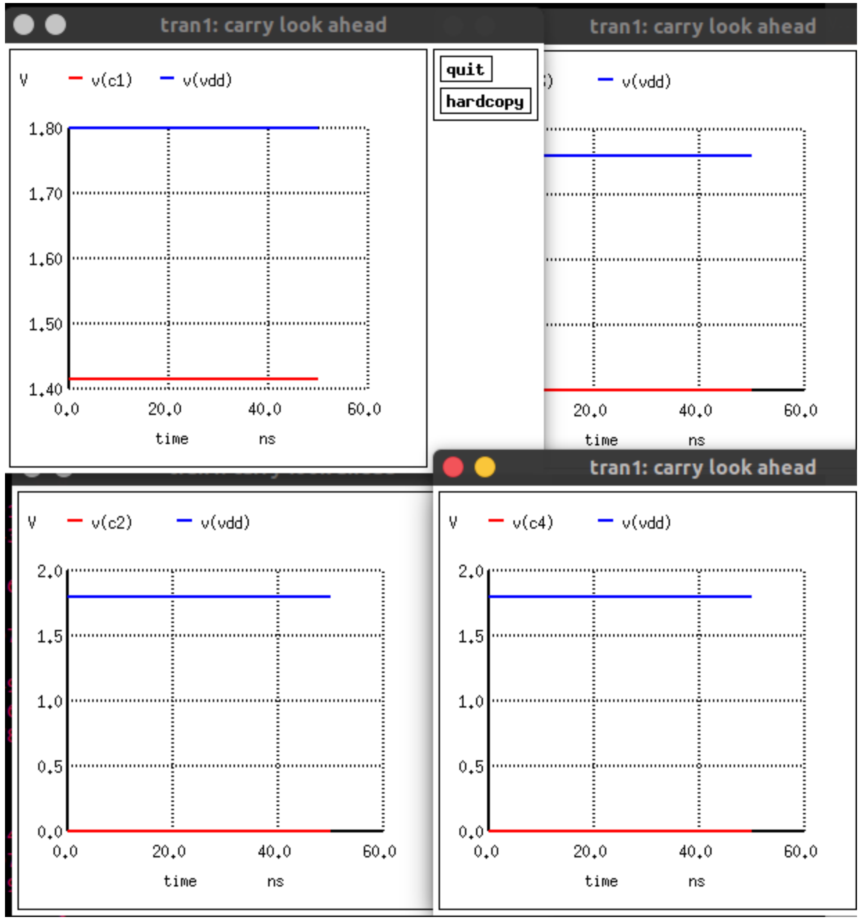


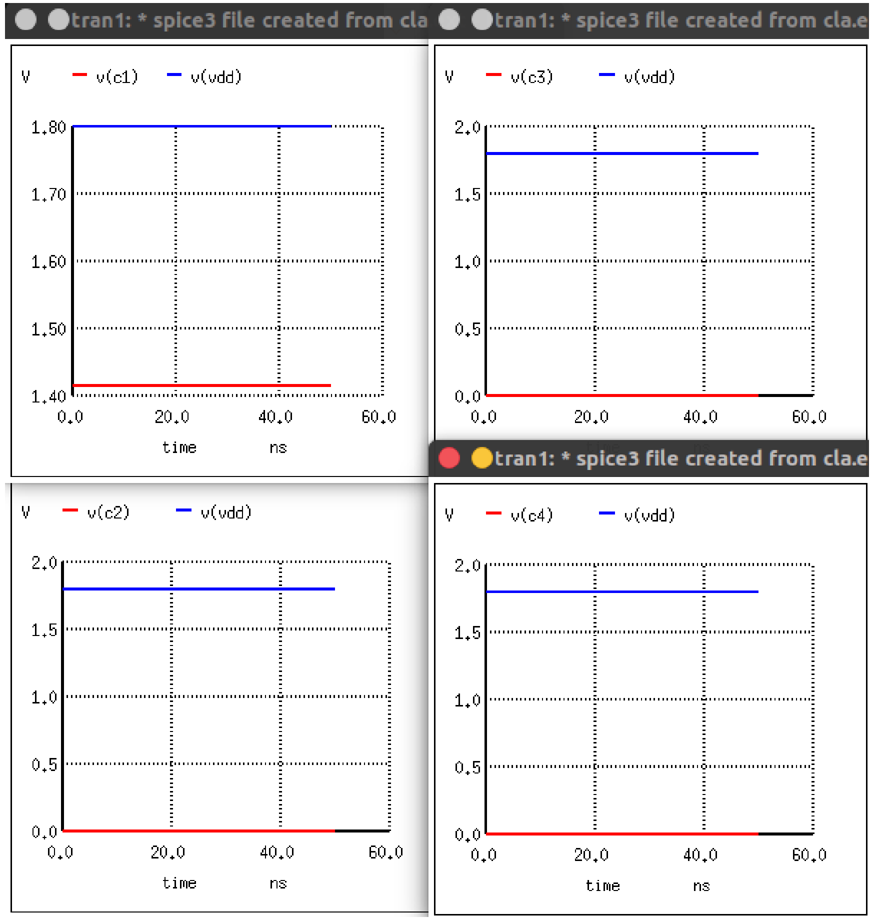
```
vp1 p1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns
vp2 p2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
vp3 p3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
vp4 p4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns

Vg1 g1 0 pulse 1.8 1.8 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg2 g2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg3 g3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vg4 g4 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
```

Plots

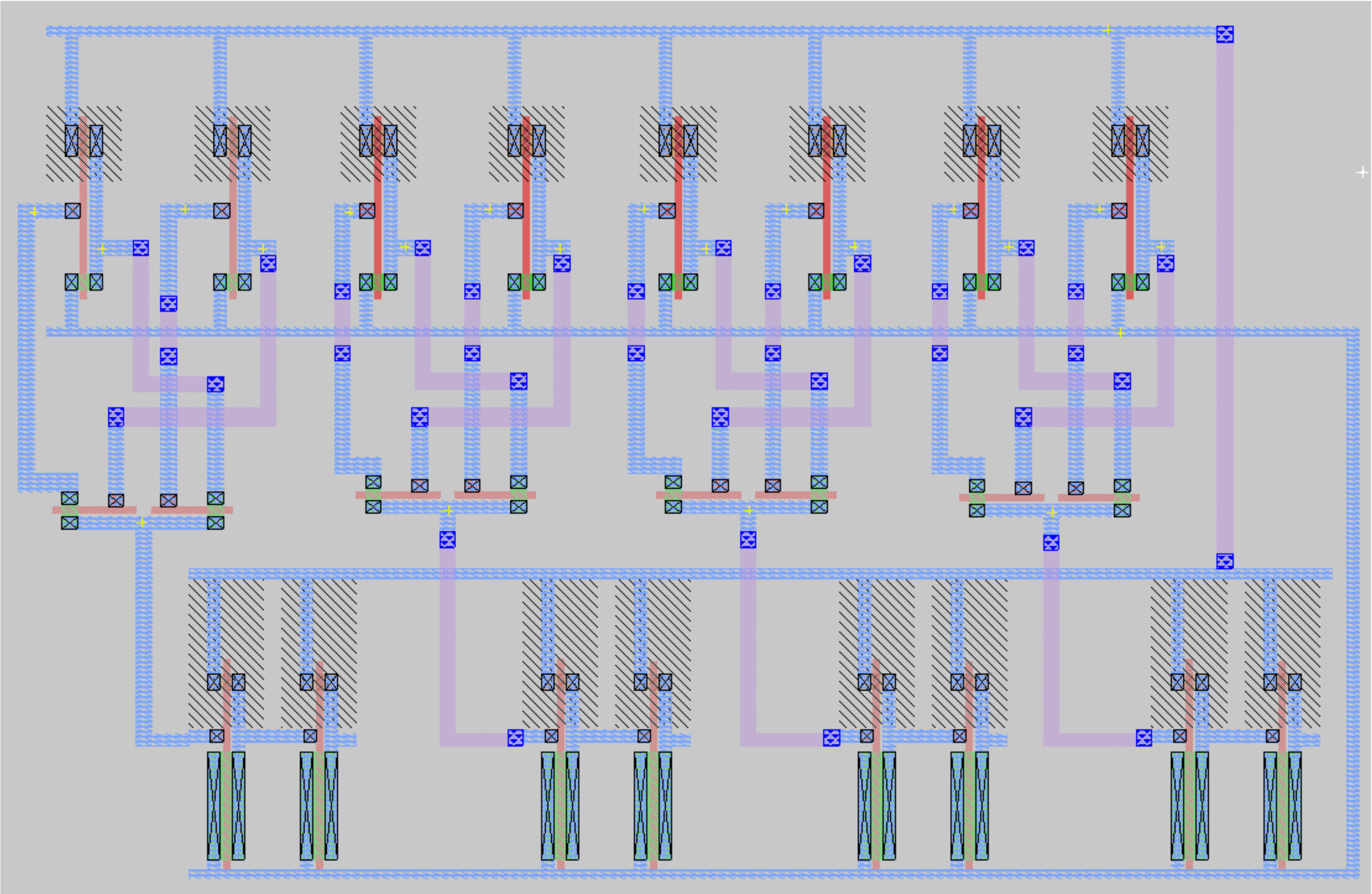
Pre-Layout

Post-Layout

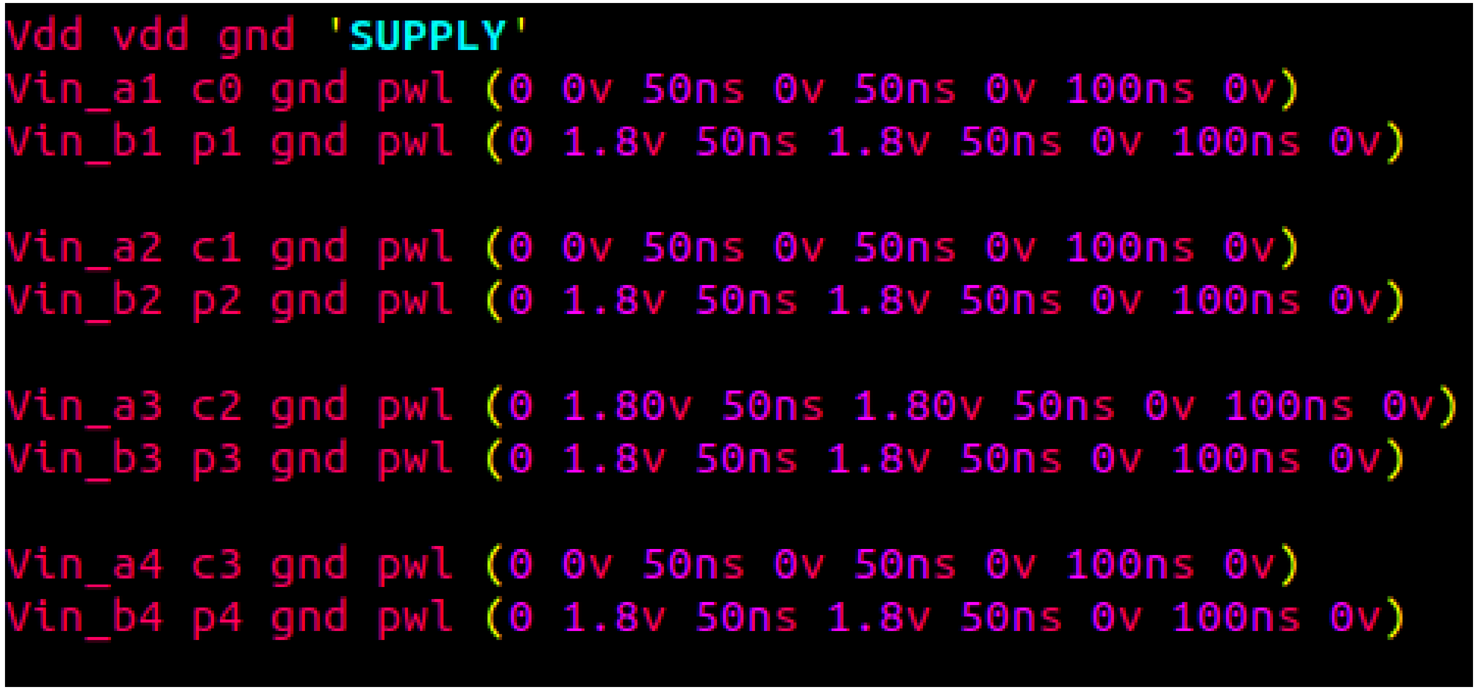



tran1: * spice3 file created from cla
tran1: * spice3 file created from cla.e
V
v(c1)
v(vdd)
1.80
1.70
1.60
1.50
1.40
0.0
20.0
40.0
60.0
time
ns
v(c3)
2.0
1.5
1.0
0.5
0.0
tran1: * spice3 file created from cla.e
v(c2)
v(c4)

Magic Layout for SUM Block



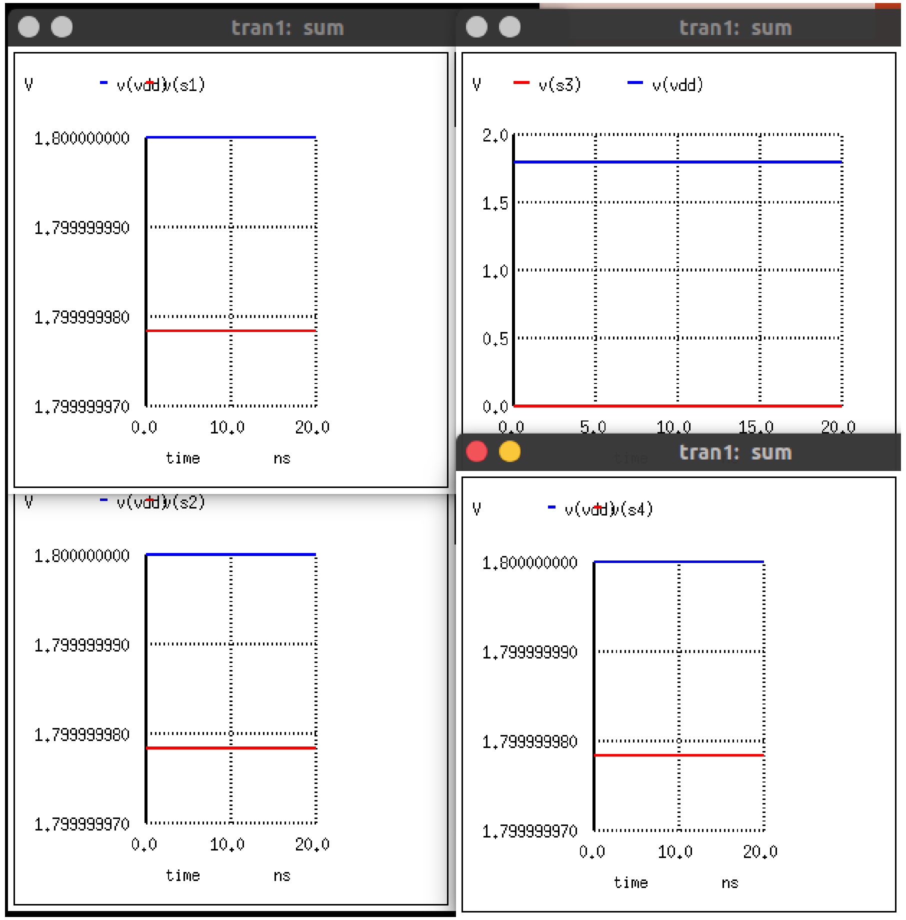
Inputs:



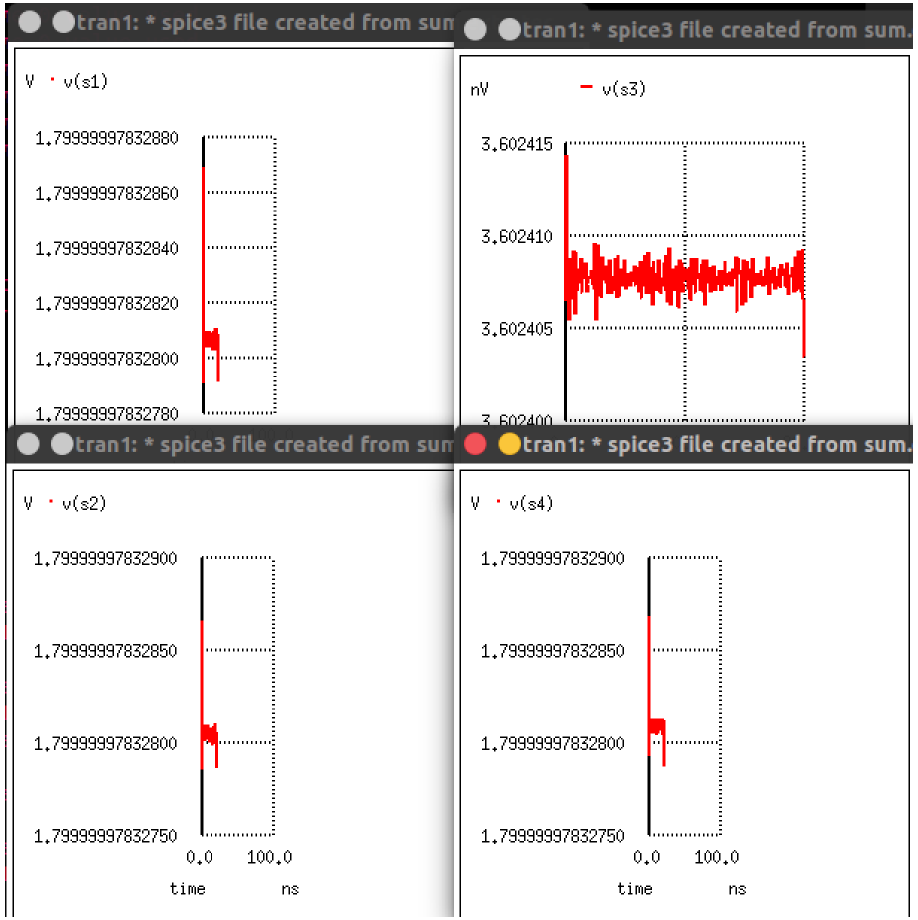
```
Vdd vdd gnd 'SUPPLY'
Vin_a1 c0 gnd pwl (0 0v 50ns 0v 50ns 0v 100ns 0v)
Vin_b1 p1 gnd pwl (0 1.8v 50ns 1.8v 50ns 0v 100ns 0v)

Vin_a2 c1 gnd pwl (0 0v 50ns 0v 50ns 0v 100ns 0v)
Vin_b2 p2 gnd pwl (0 1.8v 50ns 1.8v 50ns 0v 100ns 0v)

Vin_a3 c2 gnd pwl (0 1.80v 50ns 1.80v 50ns 0v 100ns 0v)
Vin_b3 p3 gnd pwl (0 1.8v 50ns 1.8v 50ns 0v 100ns 0v)

Vin_a4 c3 gnd pwl (0 0v 50ns 0v 50ns 0v 100ns 0v)
Vin_b4 p4 gnd pwl (0 1.8v 50ns 1.8v 50ns 0v 100ns 0v)
```

Plots

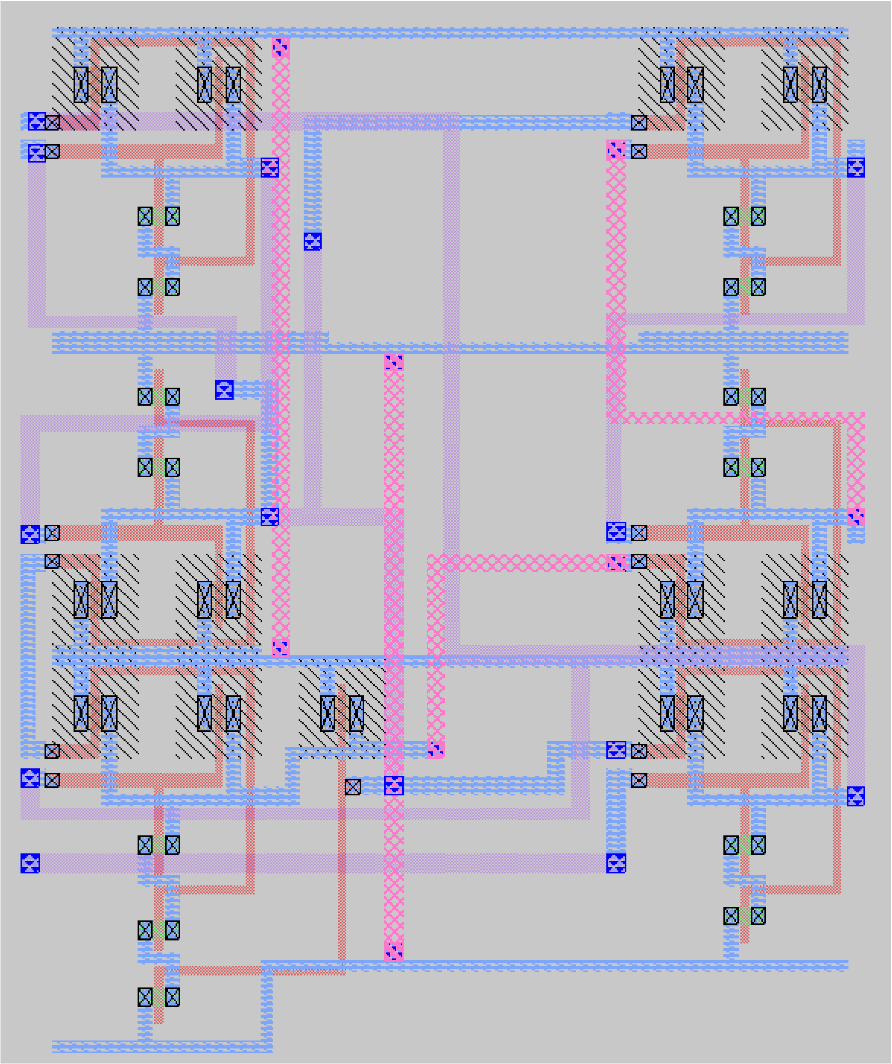
Pre-Layout



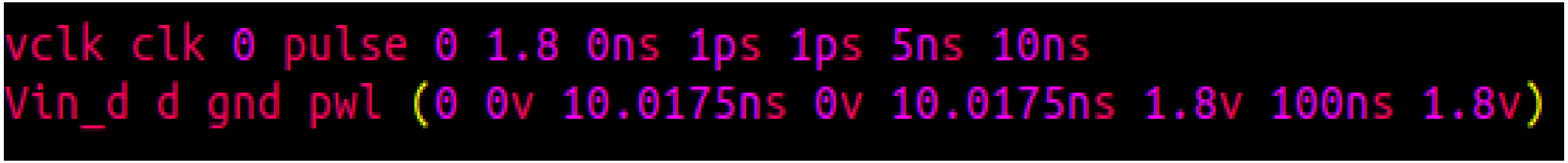
Post-Layout



Magic Layout for D FlipFlop



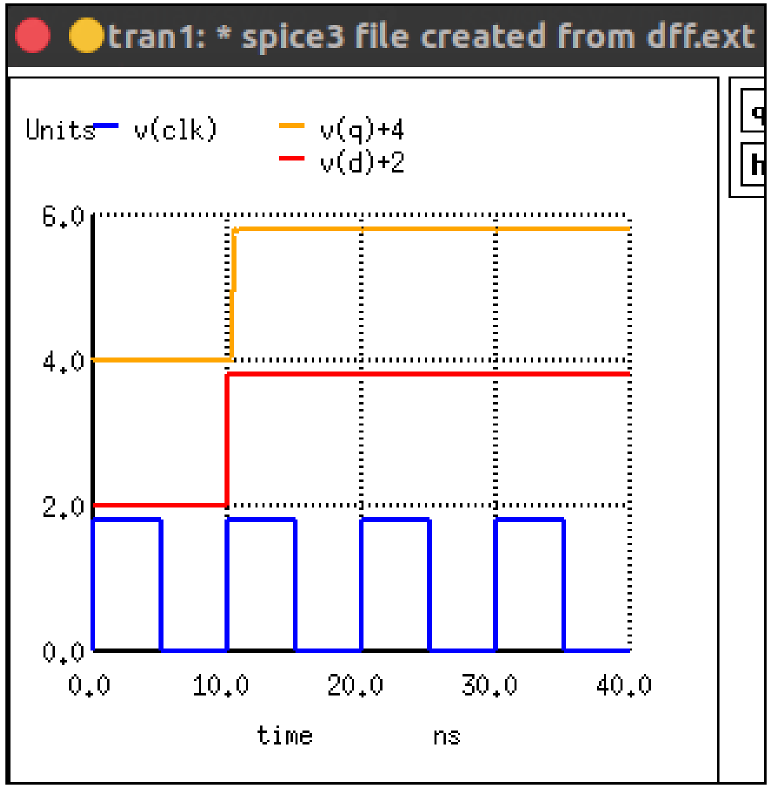
Input



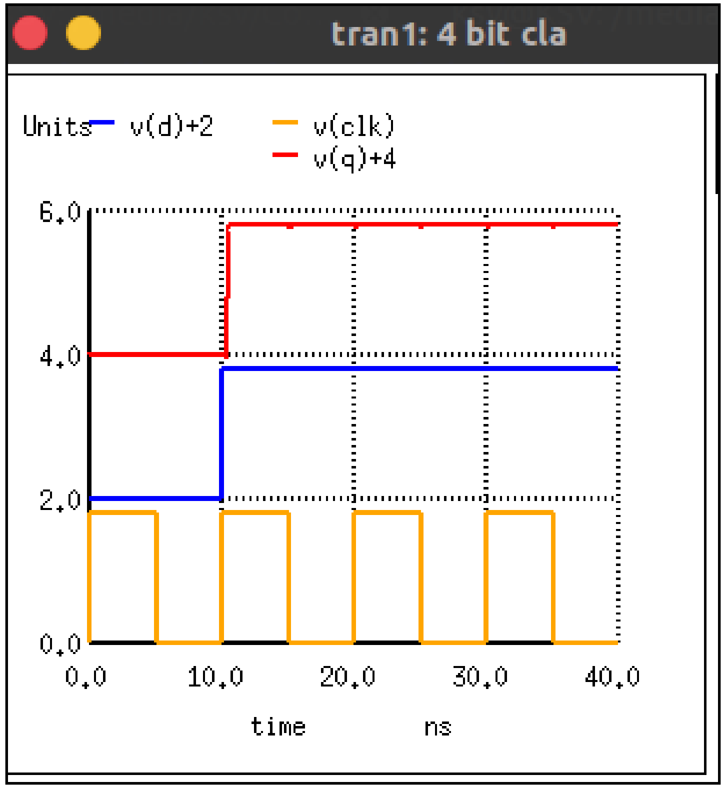
```
vclk clk 0 pulse 0 1.8 0ns 1ps 1ps 5ns 10ns
Vin_d d gnd pwl (0 0v 10.0175ns 0v 10.0175ns 1.8v 100ns 1.8v)
```

Plots

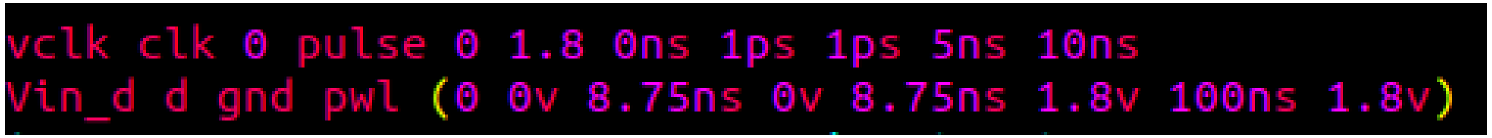
Pre-Layout



Post-Layout



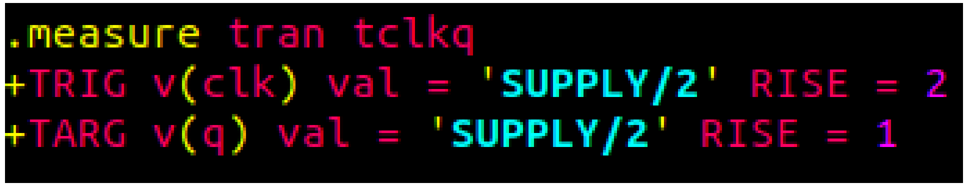
Clk to Q delay



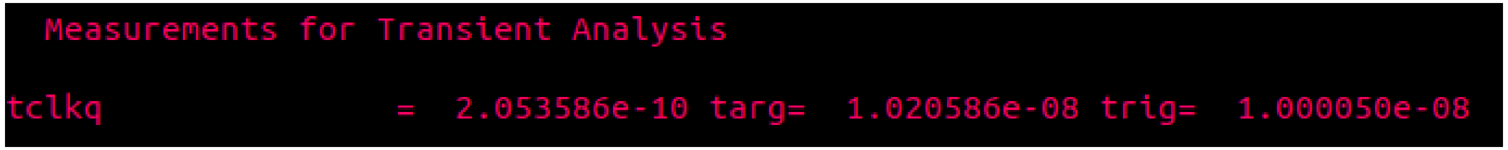
```
vclk clk 0 pulse 0 1.8 0ns 1ps 1ps 5ns 10ns
Vin_d d gnd pwl (0 0v 8.75ns 0v 8.75ns 1.8v 100ns 1.8v)
```

Clk and Input Parameters

Measurement Criteria



```
.measure tran tclkq
+TRIG v(clk) val = 'SUPPLY/2' RISE = 2
+TARG v(q) val = 'SUPPLY/2' RISE = 1
```

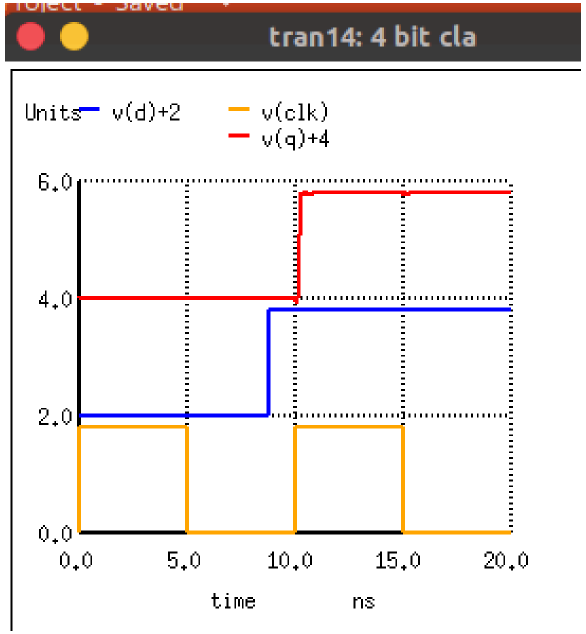


```
  Measurements for Transient Analysis

tclkq                 =  2.053586e-10 targ=  1.020586e-08 trig=  1.000050e-08
```

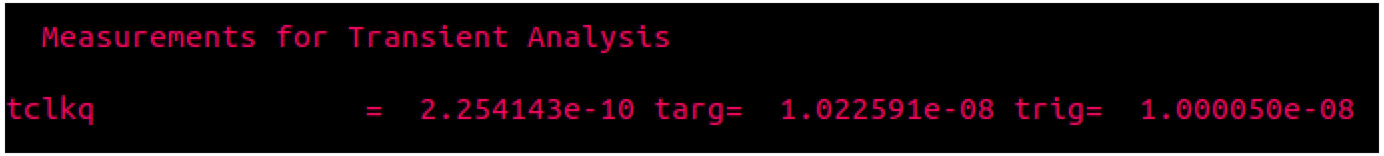
Measured Values

Obtained value of tclkq = 0.205ns.
Here we have our 2nd rising edge of clk at 10 ns.It can be noted that input is given after a delay of 8.75 which is away from the setup time that is found next

Setup Time



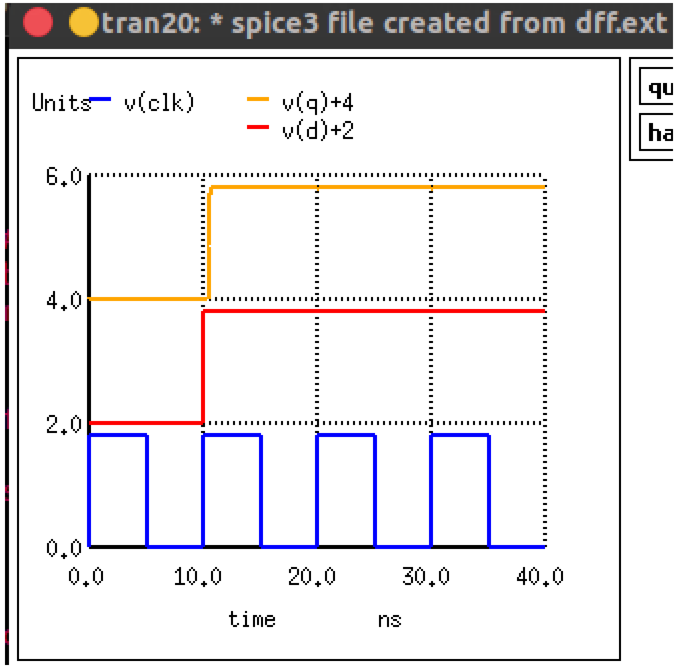
Measured Values



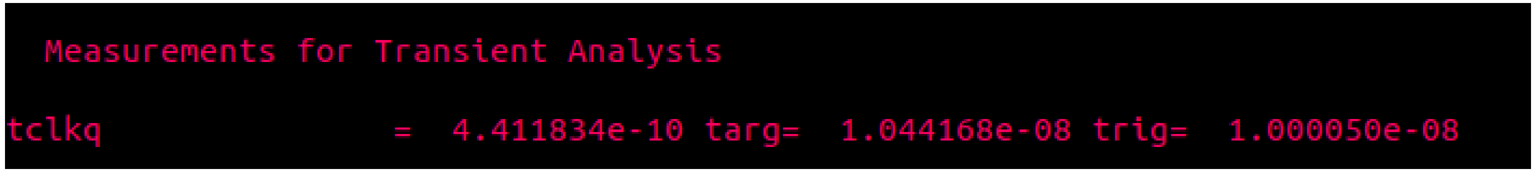
```
 Measurements for Transient Analysis

tclkq                =  2.254143e-10 targ=  1.022591e-08 trig=  1.000050e-08
```

Here input at D is applied after a delay of 9.83ns for which we observe tclkq = 0.225ns which is a rise of 0.02 ns ie. ~10% change of original value. Therefore Setup time = clk time – delay time I.e 10 – 9.83 = 0.17ns

Hold Time



Measured Values



```
 Measurements for Transient Analysis

tclkq                = 4.411834e-10 targ= 1.044168e-08 trig= 1.000050e-08
```

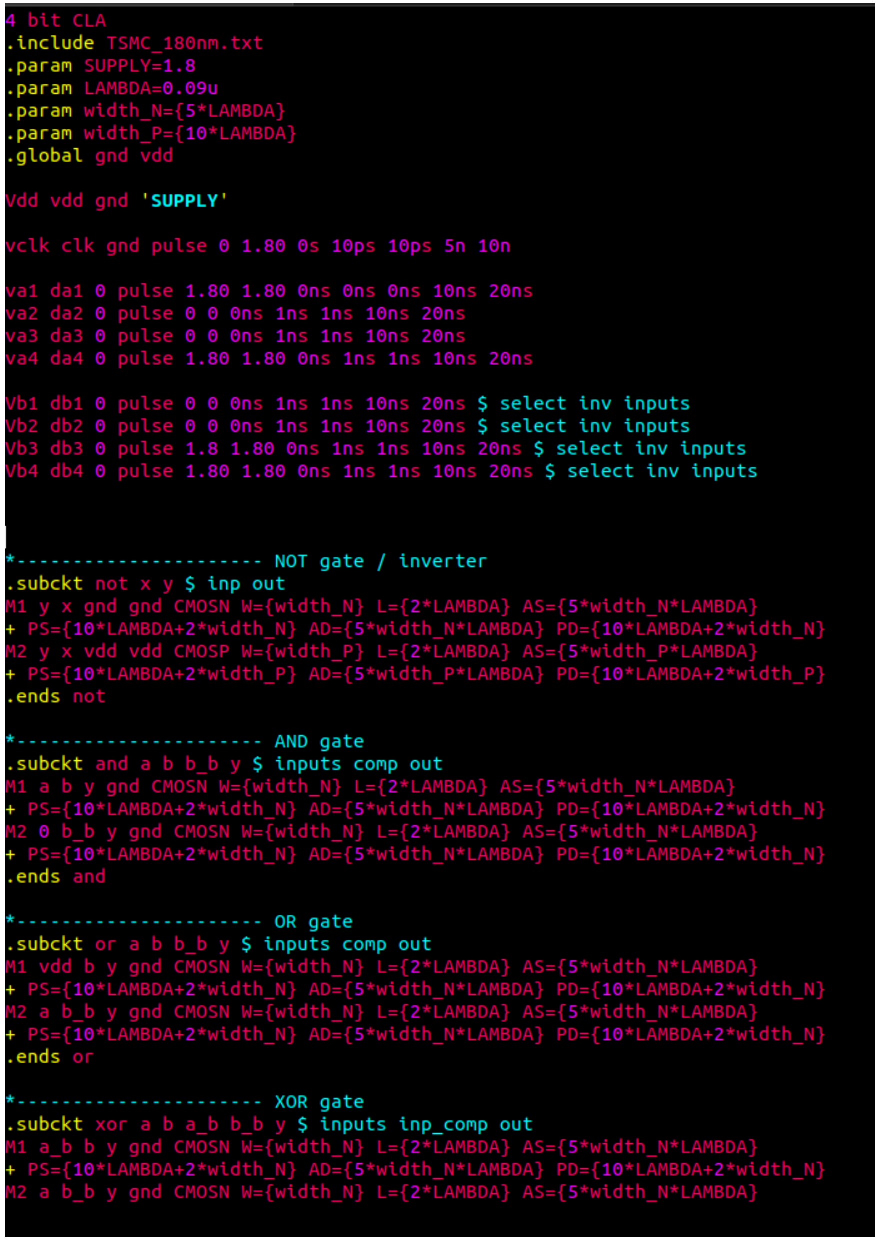
Here input at D is applied after a delay of 10.05ns~ for which we observe the output to change in the same clock cycle with a delay of 0.441ns. Therefore, Hold time = 0.05ns

- We can observe that in all previous cases for comaparison of pre-layouts and post-layouts , outputs obtained are almost same with very minor differences.
- Major difference is observed only with timings of the flip flop where we can see increase in all clk to Q, setup time and hold time compared to pre layout netlists.

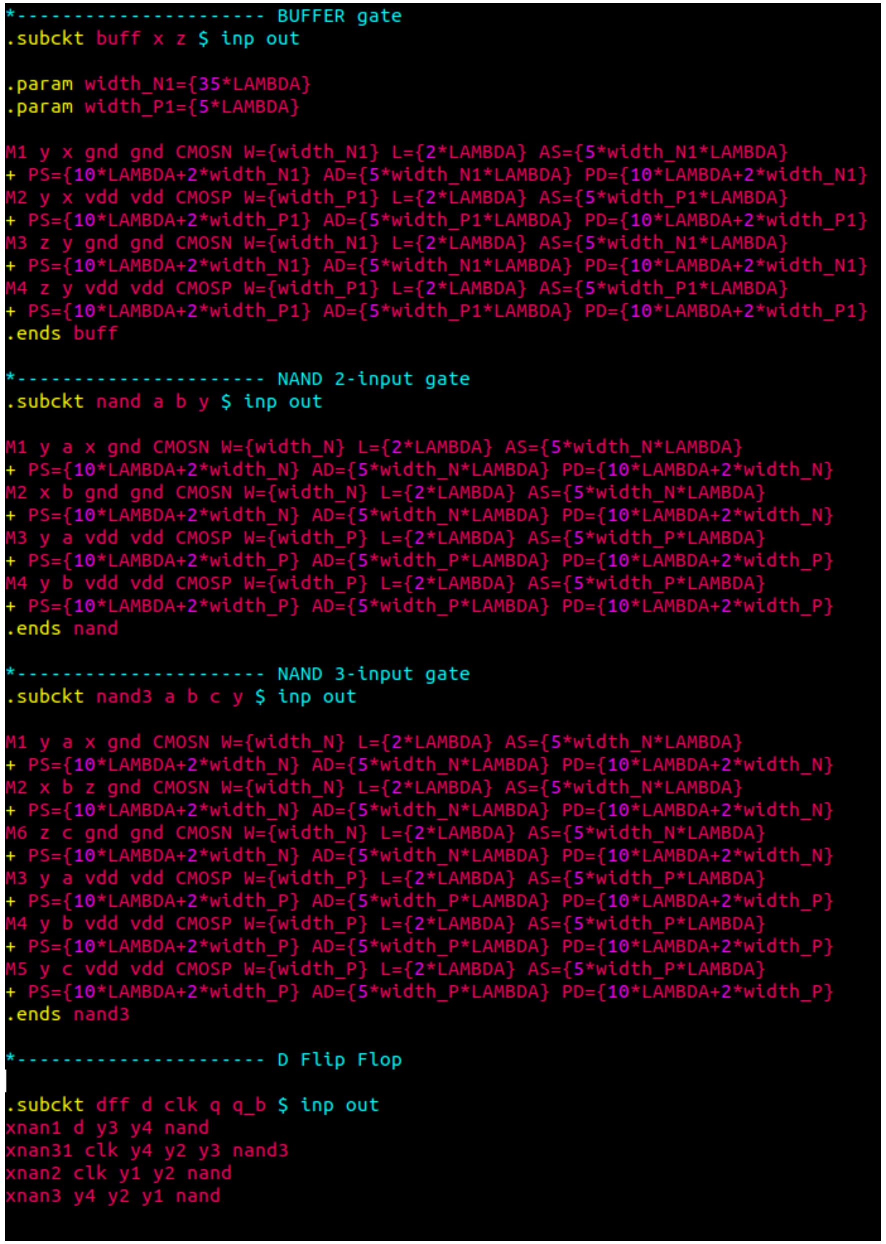
Question 7

Integrate different block designed in previous steps and write the netlist for the full circuit shown in figure 1(i). Use NGSPICE and verify the functionality of the circuit. Attach the properly annotated waveforms. Report the worst case delay of your adder and maximum clock speed for which your design operates correctly.

NETLIST

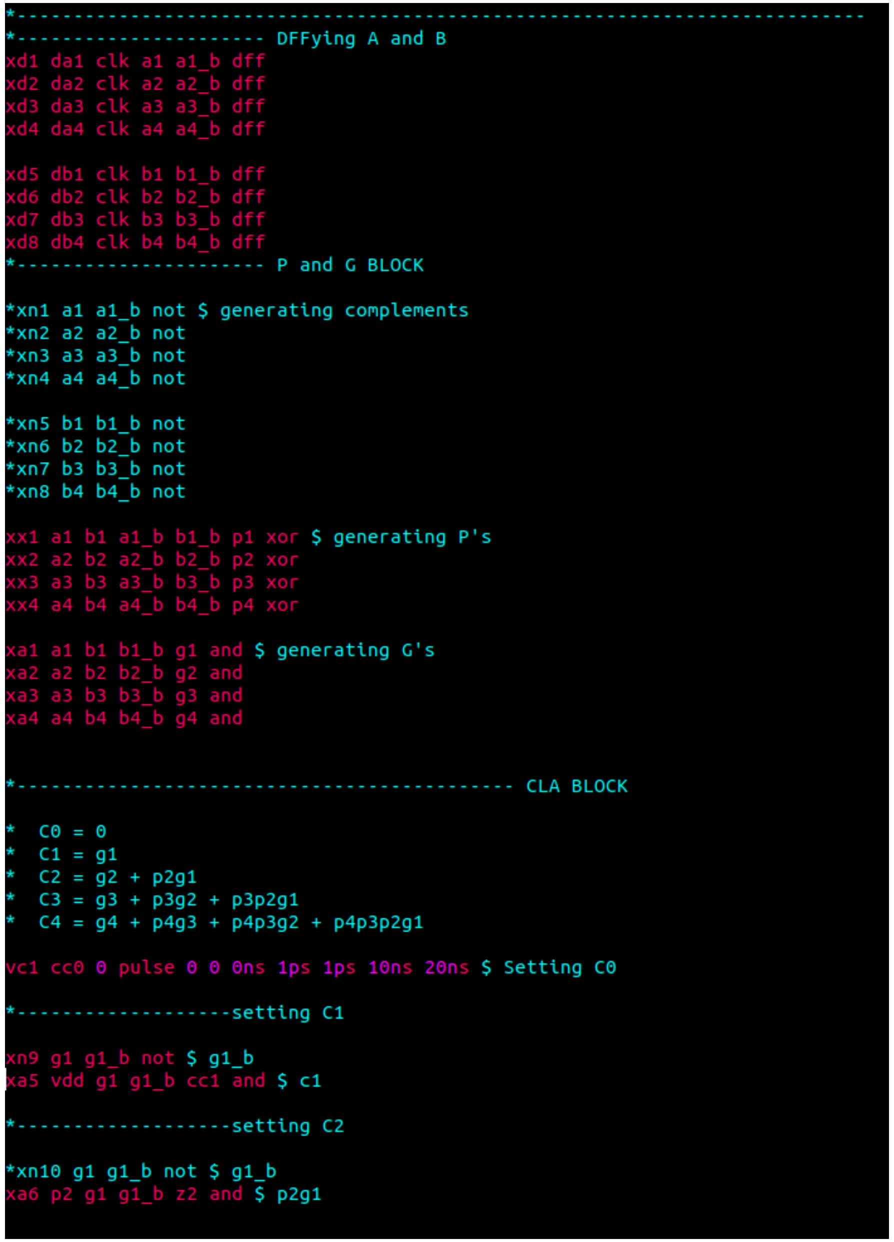


```
4 bit CLA
.include TSMC_180nm.txt
.param SUPPLY=1.8
.param LAMBDA=0.09u
.param width_N={5*LAMBDA}
.param width_P={10*LAMBDA}
.global gnd vdd

Vdd vdd gnd 'SUPPLY'

vclk clk gnd pulse 0 1.80 0s 10ps 10ps 5n 10n

va1 da1 0 pulse 1.80 1.80 0ns 0ns 0ns 10ns 20ns
va2 da2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va3 da3 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns
va4 da4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns

Vb1 db1 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb2 db2 0 pulse 0 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb3 db3 0 pulse 1.8 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb4 db4 0 pulse 1.80 1.80 0ns 1ns 1ns 10ns 20ns $ select inv inputs


*---------------------- NOT gate / inverter
.subckt not x y $ inp out
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

*---------------------- AND gate
.subckt and a b b_b y $ inputs comp out
M1 a b y 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 0 b_b y 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 and

*---------------------- OR gate
.subckt or a b b_b y $ inputs comp out
M1 vdd b y 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 b_b y 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 or

*---------------------- XOR gate
.subckt xor a b a_b b_b y $ inputs inp_comp out
M1 a_b b y 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 b_b y gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
```

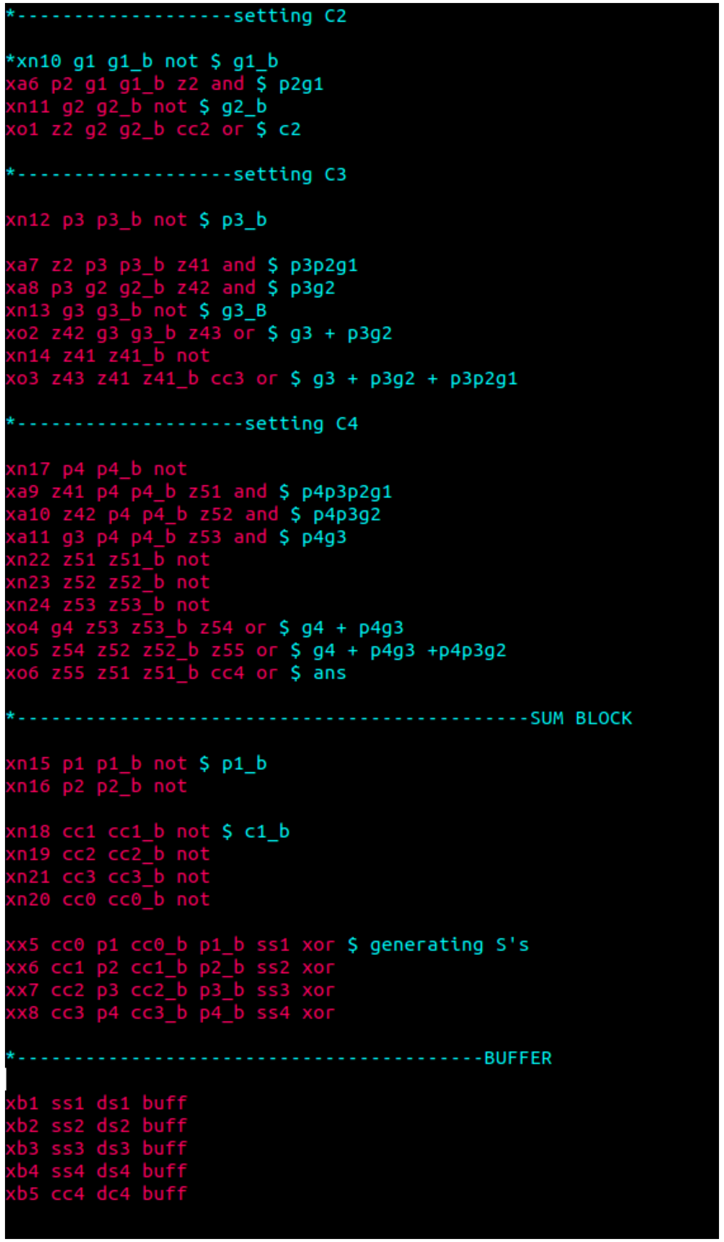


```
*---------------------- BUFFER gate
.subckt buff x z $ inp out

.param width_N1={35*LAMBDA}
.param width_P1={5*LAMBDA}

M1 y x gnd gnd CMOSN W={width_N1} L={2*LAMBDA} AS={5*width_N1*LAMBDA}
+ PS={10*LAMBDA+2*width_N1} AD={5*width_N1*LAMBDA} PD={10*LAMBDA+2*width_N1}
M2 y x vdd vdd CMOSP W={width_P1} L={2*LAMBDA} AS={5*width_P1*LAMBDA}
+ PS={10*LAMBDA+2*width_P1} AD={5*width_P1*LAMBDA} PD={10*LAMBDA+2*width_P1}
M3 z y gnd gnd CMOSN W={width_N1} L={2*LAMBDA} AS={5*width_N1*LAMBDA}
+ PS={10*LAMBDA+2*width_N1} AD={5*width_N1*LAMBDA} PD={10*LAMBDA+2*width_N1}
M4 z y vdd vdd CMOSP W={width_P1} L={2*LAMBDA} AS={5*width_P1*LAMBDA}
+ PS={10*LAMBDA+2*width_P1} AD={5*width_P1*LAMBDA} PD={10*LAMBDA+2*width_P1}
.ends buff

*---------------------- NAND 2-input gate
.subckt nand a b y $ inp out

M1 y a x gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 x 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}
M3 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}
M4 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}
.ends nand

*---------------------- NAND 3-input gate
.subckt nand3 a b c y $ inp out

M1 y a x gnd CMOSN W={width_N} L={2*LAMBDA} AS={5*width_N*LAMBDA}
+ PS={10*LAMBDA+2*width_N} AD={5*width_N*LAMBDA} PD={10*LAMBDA+2*width_N}
M2 x b z 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 z 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}
M3 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}
M4 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}
M5 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}
.ends nand3

*---------------------- D Flip Flop

.subckt dff d clk q q_b $ inp out
xnan1 d y3 y4 nand
xnan31 clk y4 y2 y3 nand3
xnan2 clk y1 y2 nand
xnan3 y4 y2 y1 nand
```

NETLIST

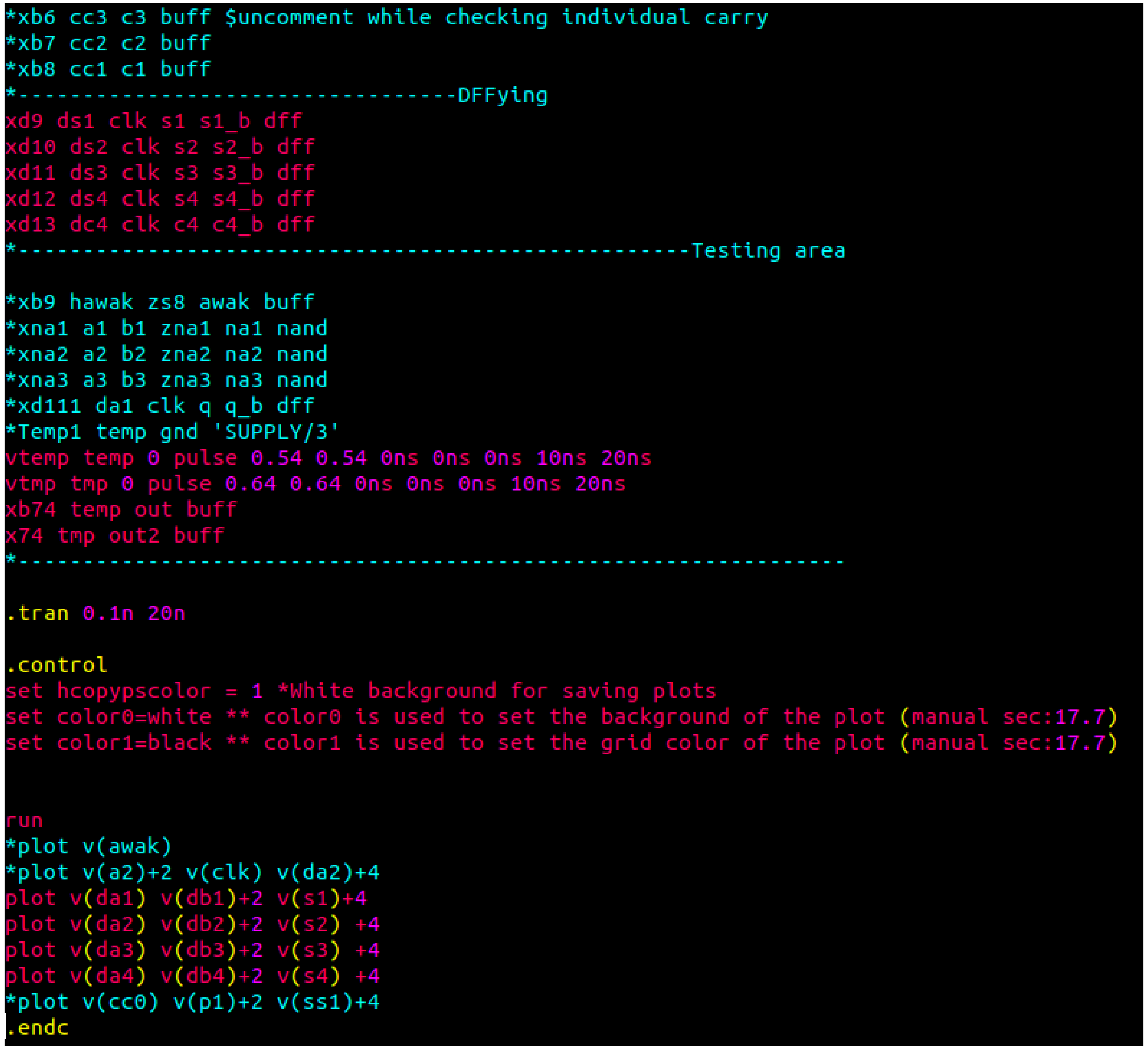


```
*---------------------------------------------------------------------------
*---------------------- DFFying A and B
xd1 da1 clk a1 a1_b dff
xd2 da2 clk a2 a2_b dff
xd3 da3 clk a3 a3_b dff
xd4 da4 clk a4 a4_b dff

xd5 db1 clk b1 b1_b dff
xd6 db2 clk b2 b2_b dff
xd7 db3 clk b3 b3_b dff
xd8 db4 clk b4 b4_b dff
*---------------------- P and G BLOCK

*xn1 a1 a1_b not $ generating complements
*xn2 a2 a2_b not
*xn3 a3 a3_b not
*xn4 a4 a4_b not

*xn5 b1 b1_b not
*xn6 b2 b2_b not
*xn7 b3 b3_b not
*xn8 b4 b4_b not

xx1 a1 b1 a1_b b1_b p1 xor $ generating P's
xx2 a2 b2 a2_b b2_b p2 xor
xx3 a3 b3 a3_b b3_b p3 xor
xx4 a4 b4 a4_b b4_b p4 xor

xa1 a1 b1 b1_b g1 and $ generating G's
xa2 a2 b2 b2_b g2 and
xa3 a3 b3 b3_b g3 and
xa4 a4 b4 b4_b g4 and


*-------------------------------------------- CLA BLOCK

*  C0 = 0
*  C1 = g1
*  C2 = g2 + p2g1
*  C3 = g3 + p3g2 + p3p2g1
*  C4 = g4 + p4g3 + p4p3g2 + p4p3p2g1

vc1 cc0 0 pulse 0 0 0ns 1ps 1ps 10ns 20ns $ Setting C0

*-------------------setting C1

xn9 g1 g1_b not $ g1_b
xa5 vdd g1 g1_b cc1 and $ c1

*-------------------setting C2

*xn10 g1 g1_b not $ g1_b
xa6 p2 g1 g1_b z2 and $ p2g1
```



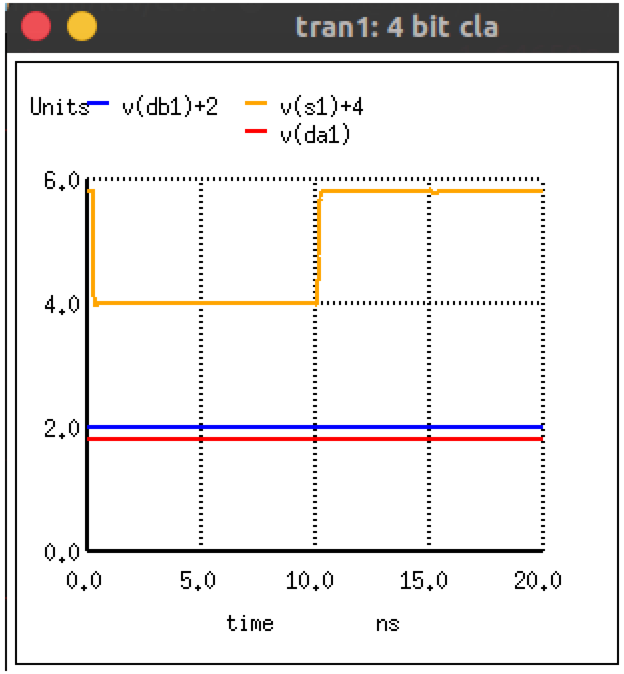
```
*-------------------setting C2

*xn10 g1 g1_b not $ g1_b
xa6 p2 g1 g1_b z2 and $ p2g1
xn11 g2 g2_b not $ g2_b
xo1 z2 g2 g2_b cc2 or $ c2

*-------------------setting C3

xn12 p3 p3_b not $ p3_b

xa7 z2 p3 p3_b z41 and $ p3p2g1
xa8 p3 g2 g2_b z42 and $ p3g2
xn13 g3 g3_b not $ g3_B
xo2 z42 g3 g3_b z43 or $ g3 + p3g2
xn14 z41 z41_b not
xo3 z43 z41 z41_b cc3 or $ g3 + p3g2 + p3p2g1

*--------------------setting C4

xn17 p4 p4_b not
xa9 z41 p4 p4_b z51 and $ p4p3p2g1
xa10 z42 p4 p4_b z52 and $ p4p3g2
xa11 g3 p4 p4_b z53 and $ p4g3
xn22 z51 z51_b not
xn23 z52 z52_b not
xn24 z53 z53_b not
xo4 g4 z53 z53_b z54 or $ g4 + p4g3
xo5 z54 z52 z52_b z55 or $ g4 + p4g3 +p4p3g2
xo6 z55 z51 z51_b cc4 or $ ans

*---------------------------------------------SUM BLOCK

xn15 p1 p1_b not $ p1_b
xn16 p2 p2_b not

xn18 cc1 cc1_b not $ c1_b
xn19 cc2 cc2_b not
xn21 cc3 cc3_b not
xn20 cc0 cc0_b not

xx5 cc0 p1 cc0_b p1_b ss1 xor $ generating S's
xx6 cc1 p2 cc1_b p2_b ss2 xor
xx7 cc2 p3 cc2_b p3_b ss3 xor
xx8 cc3 p4 cc3_b p4_b ss4 xor

*-----------------------------------------BUFFER

xb1 ss1 ds1 buff
xb2 ss2 ds2 buff
xb3 ss3 ds3 buff
xb4 ss4 ds4 buff
xb5 cc4 dc4 buff
```

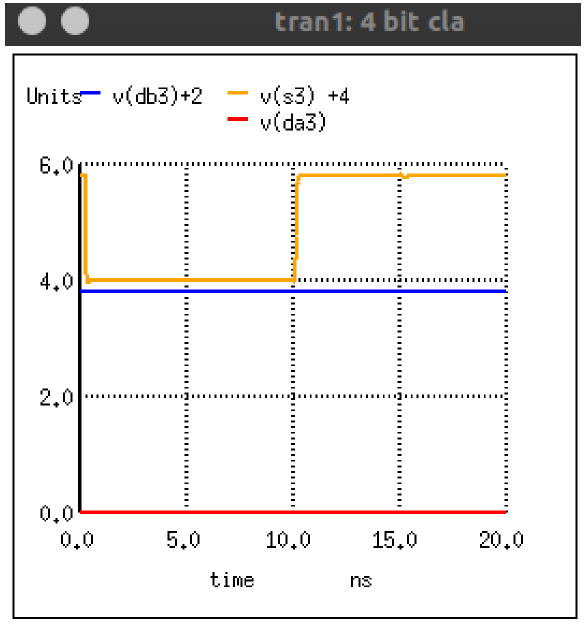
NETLIST

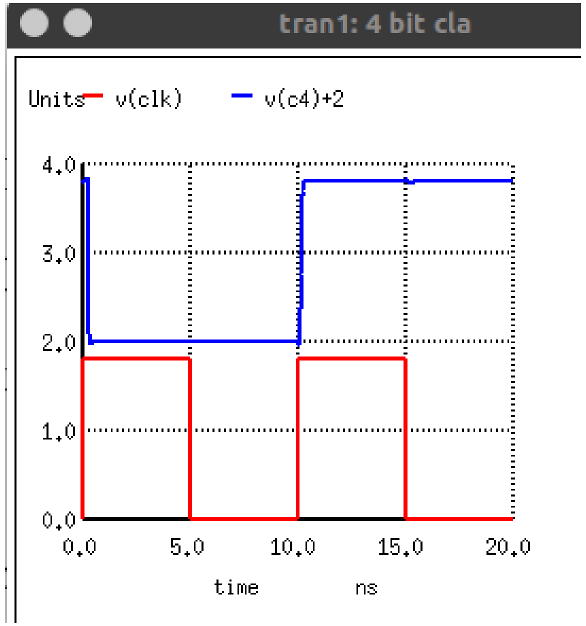


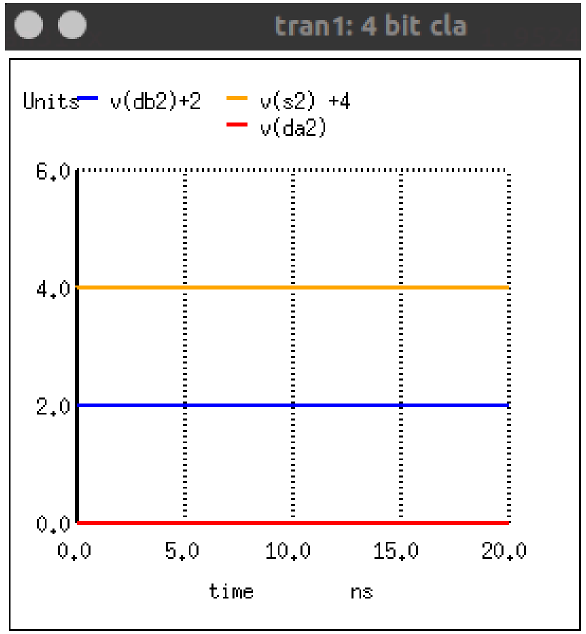
```
*xb6 cc3 c3 buff $uncomment while checking individual carry
*xb7 cc2 c2 buff
*xb8 cc1 c1 buff
*----------------------------------DFFying
xd9 ds1 clk s1 s1_b dff
xd10 ds2 clk s2 s2_b dff
xd11 ds3 clk s3 s3_b dff
xd12 ds4 clk s4 s4_b dff
xd13 dc4 clk c4 c4_b dff
*----------------------------------------------------Testing area

*xb9 hawak zs8 awak buff
*xna1 a1 b1 zna1 na1 nand
*xna2 a2 b2 zna2 na2 nand
*xna3 a3 b3 zna3 na3 nand
*xd111 da1 clk q q_b dff
*Temp1 temp gnd 'SUPPLY/3'
vtemp temp 0 pulse 0.54 0.54 0ns 0ns 0ns 10ns 20ns
vtmp tmp 0 pulse 0.64 0.64 0ns 0ns 0ns 10ns 20ns
xb74 temp out buff
x74 tmp out2 buff
*----------------------------------------------------------------

.tran 0.1n 20n

.control
set hcopypscolor = 1 *White background for saving plots
set color0=white ** color0 is used to set the background of the plot (manual sec:17.7)
set color1=black ** color1 is used to set the grid color of the plot (manual sec:17.7)


run
*plot v(awak)
*plot v(a2)+2 v(clk) v(da2)+4
plot v(da1) v(db1)+2 v(s1)+4
plot v(da2) v(db2)+2 v(s2) +4
plot v(da3) v(db3)+2 v(s3) +4
plot v(da4) v(db4)+2 v(s4) +4
*plot v(cc0) v(p1)+2 v(ss1)+4
.endc
```

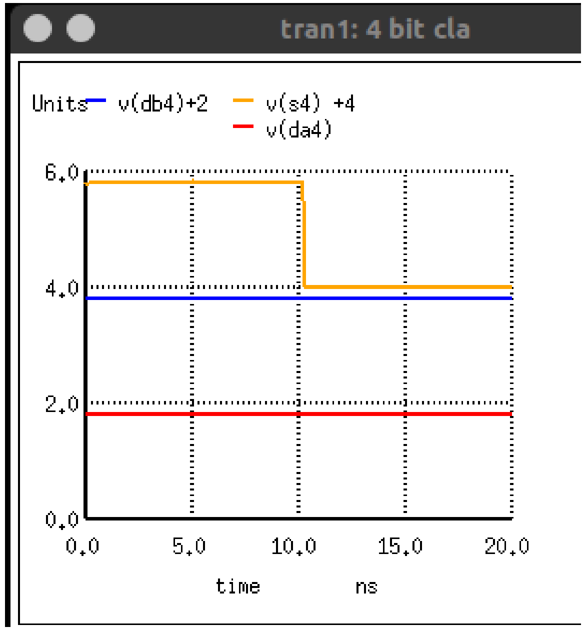
Plots





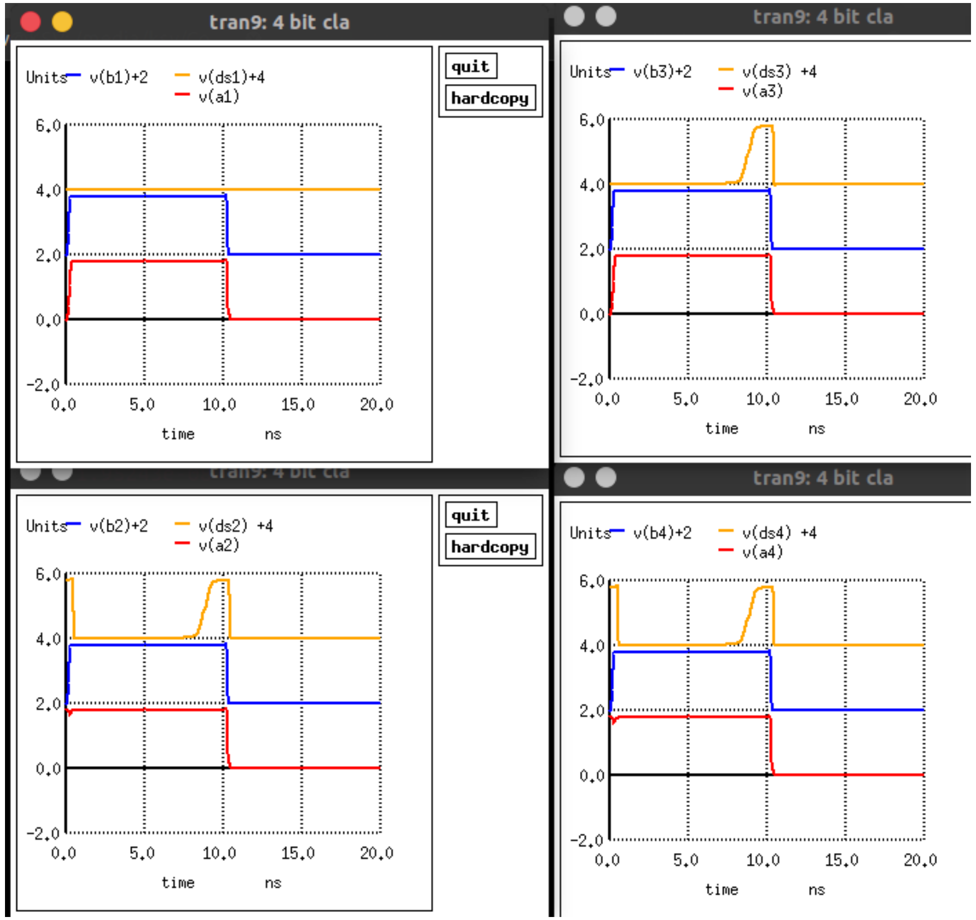


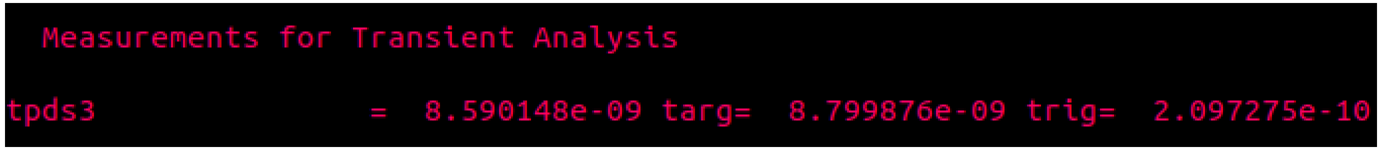




Here da, db are inputs to flipflops while s and c are output sum and carry. Clk is the clock

Worst Case Delay





```
 Measurements for Transient Analysis

tpds3               =  8.590148e-09 targ=  8.799876e-09 trig=  2.097275e-10
```

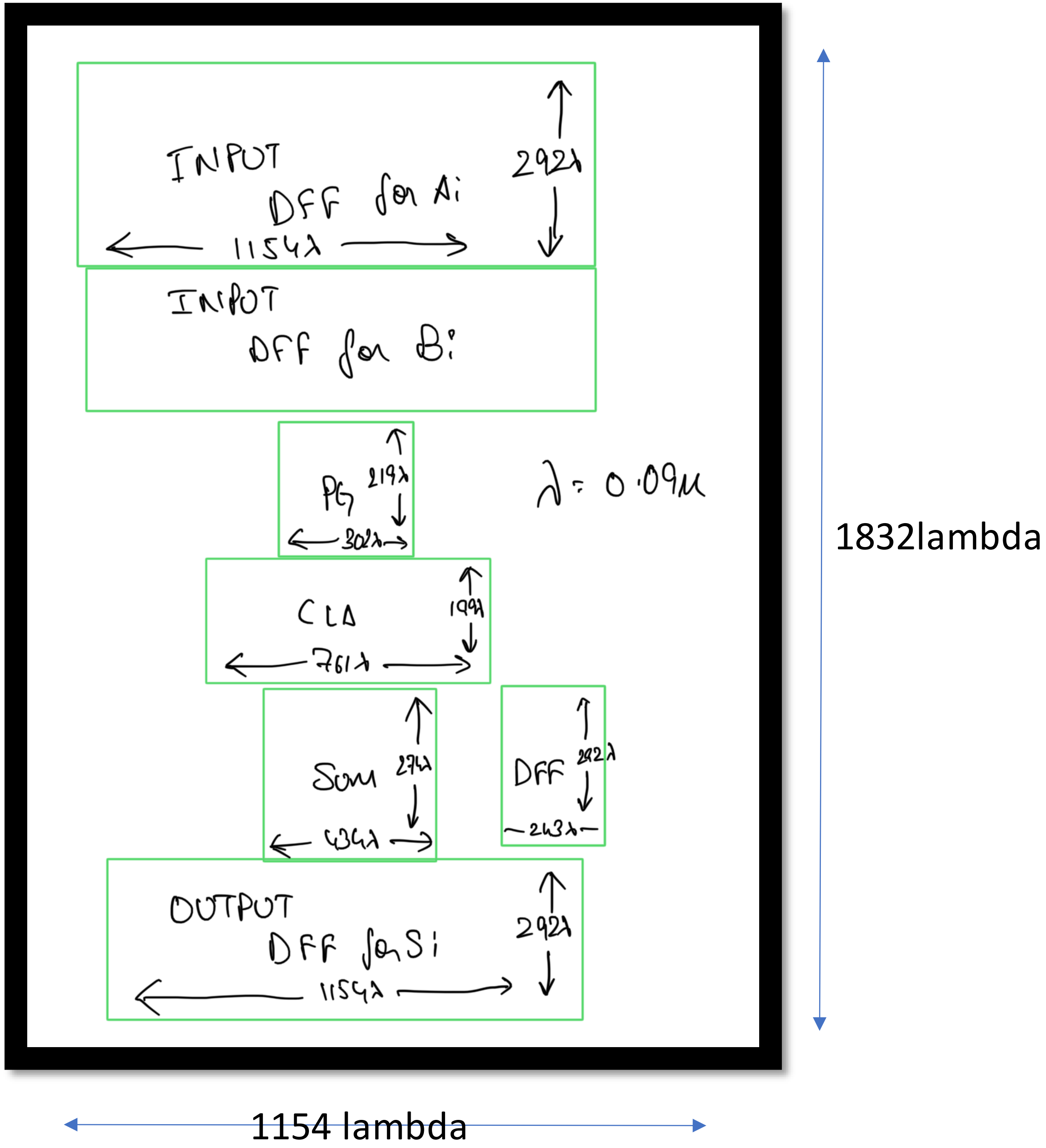
T_{clock} >= T_{clkQ} + T_{delay}+ T_{setup}. Here anyways T_{delay} >> other quantities. So T_{clock} ~= T_{delay} = 8.59nS

F_{clock} = 0.1164GHz

Question 8

Give the floor plan of the layout for the complete circuit. Identify the horizontal and vertical pitches in the regular structures.

The floorplan estimates the area of major units in the chip and defines their relative placements. The floorplan is essential to determine whether a proposed design will fit in the chip area budgeted and to estimate wiring lengths and wiring congestion.



1832lambda

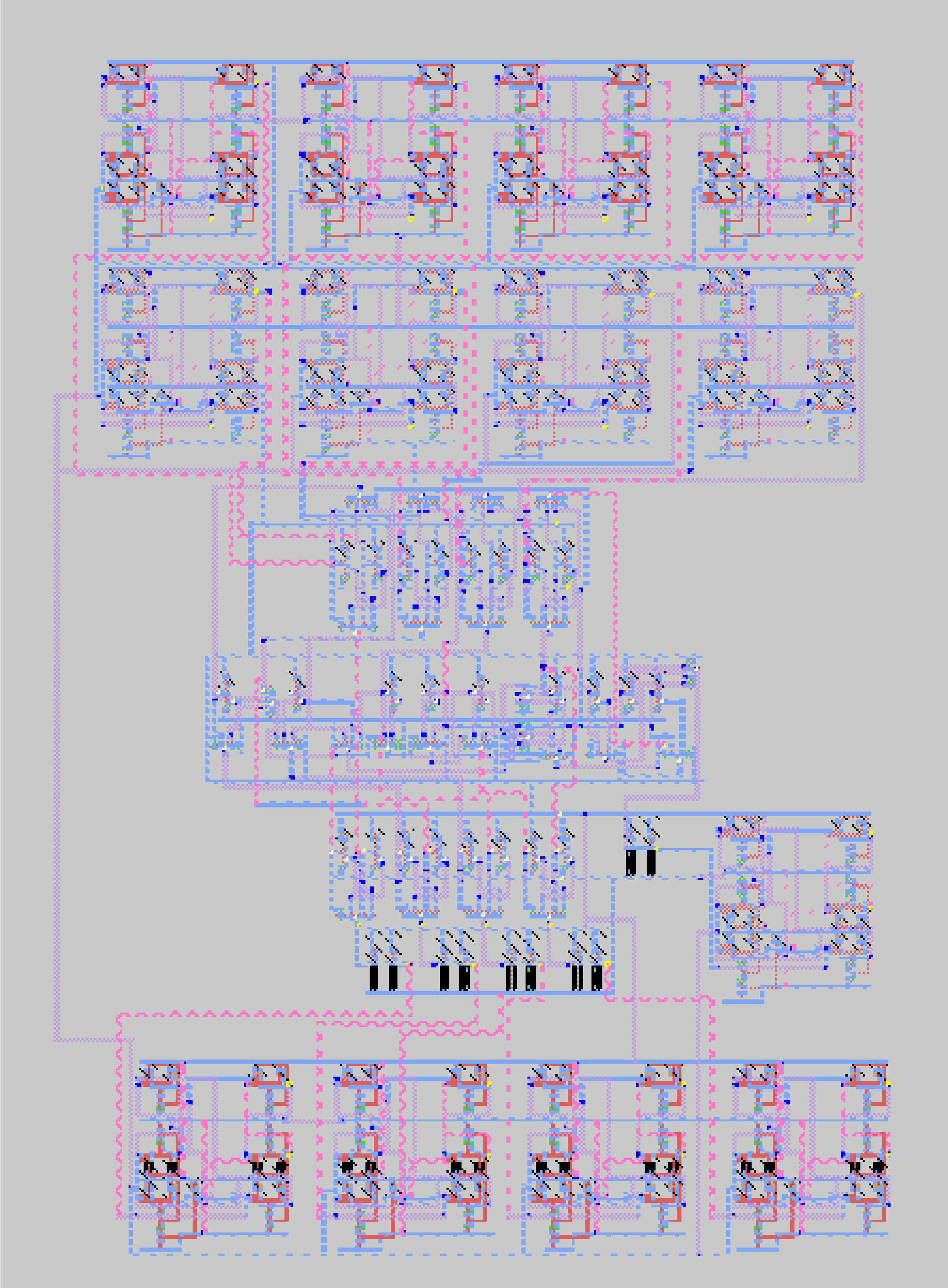
1154 lambda

Question 9 and 10

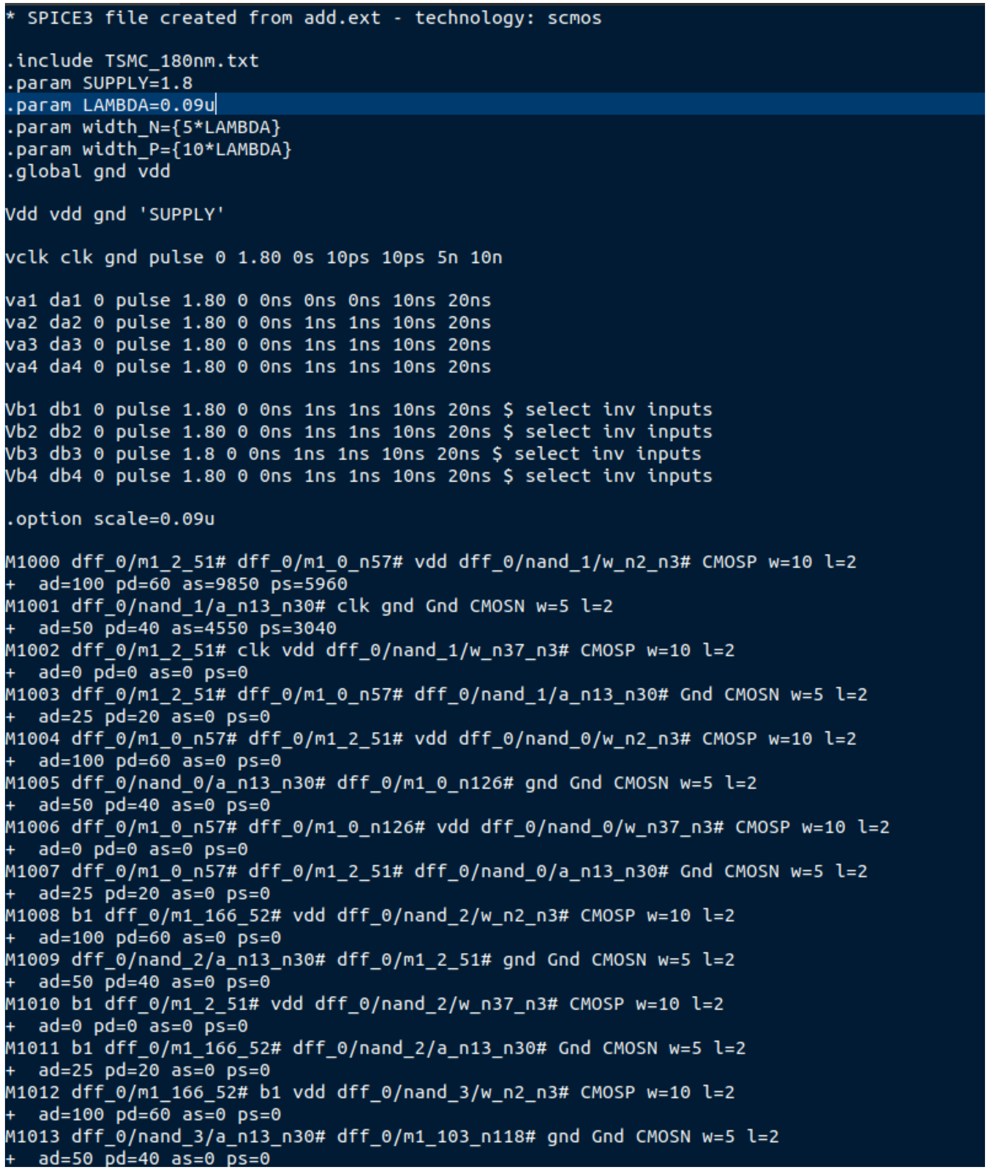
Make layout of the complete circuit and extract the spice netlist. Repeat the simulations discussed above on the extracted netlist. Give a table comparing the schematic and postlayout simulation results.

Report the delay of the CLA-adder and maximum clock frequency at which your circuit operates reliably.

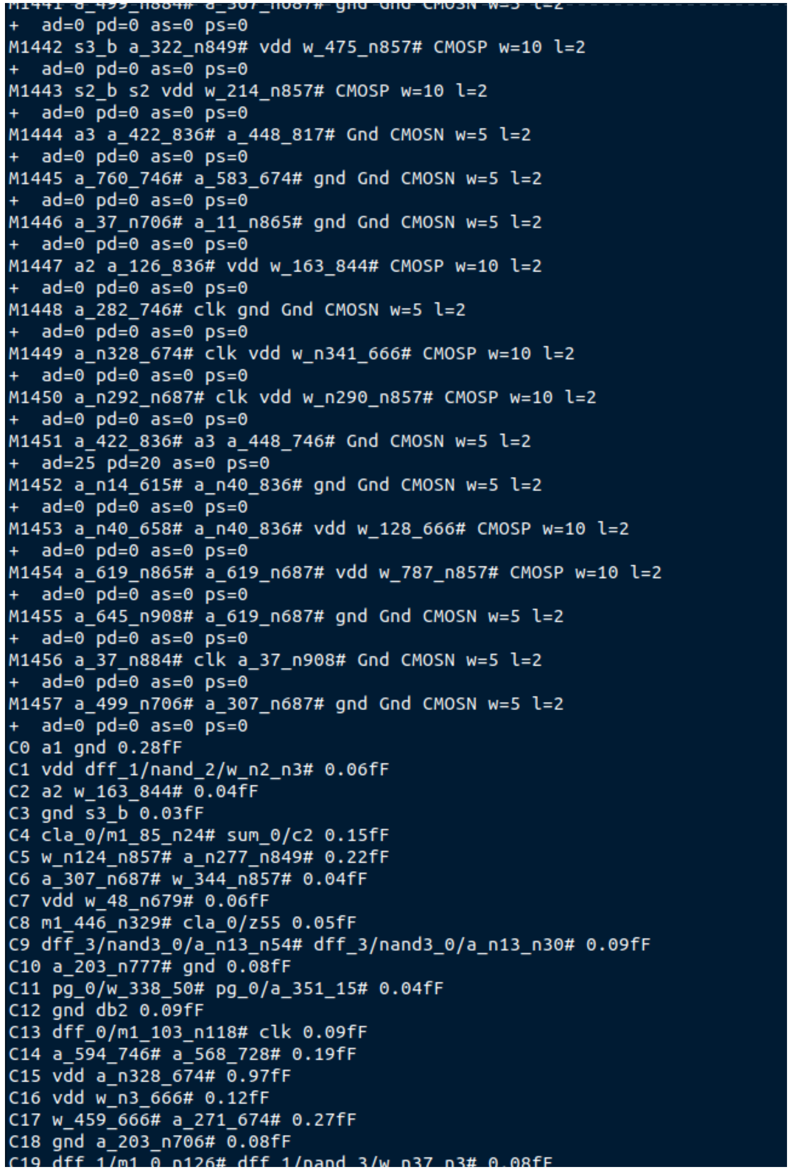
Magic Layout of Integrated circuit



NETLIST



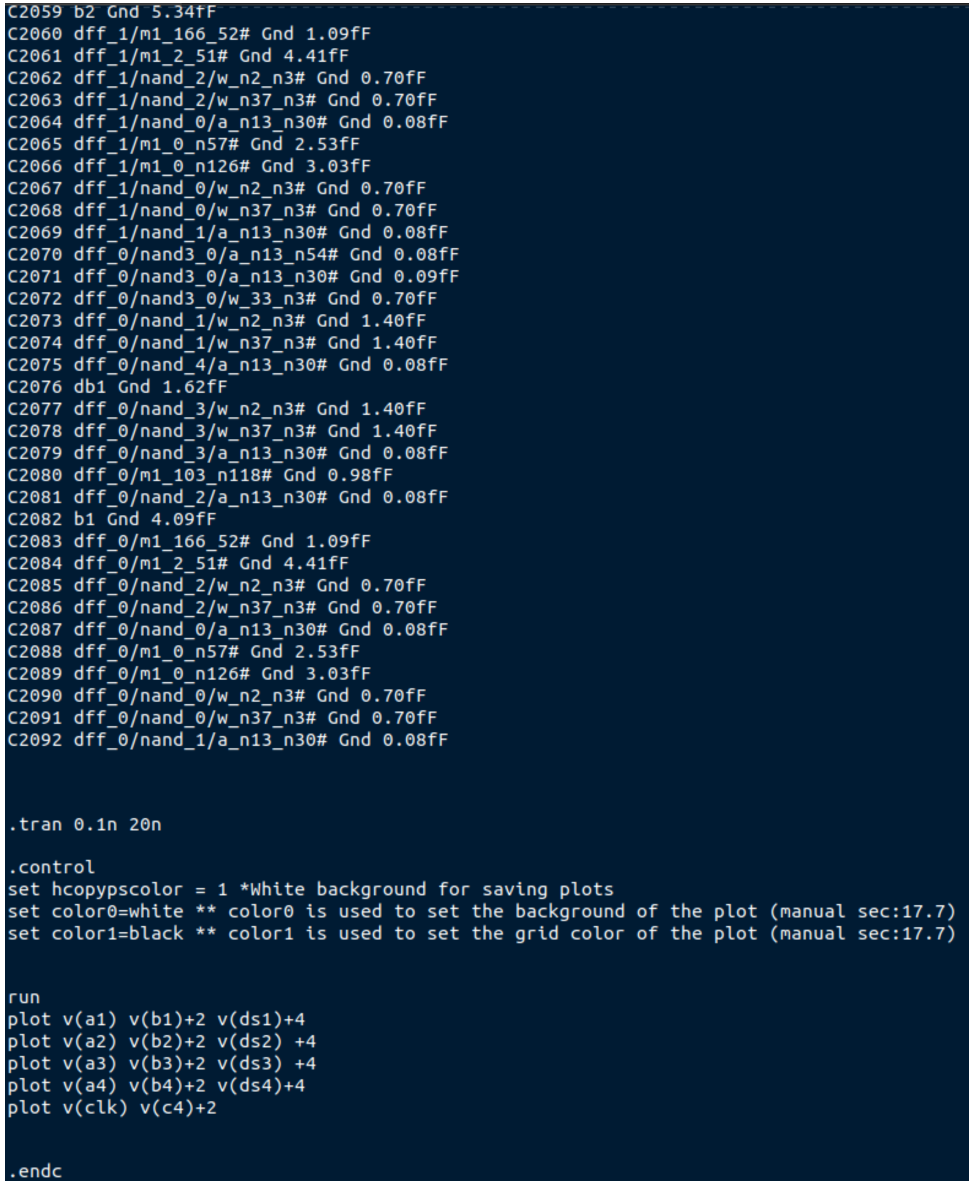
```
* SPICE3 file created from add.ext - technology: scmos

.include TSMC_180nm.txt
.param SUPPLY=1.8
.param LAMBDA=0.09u
.param width_N={5*LAMBDA}
.param width_P={10*LAMBDA}
.global gnd vdd

Vdd vdd gnd 'SUPPLY'

vclk clk gnd pulse 0 1.80 0s 10ps 10ps 5n 10n

va1 da1 0 pulse 1.80 0 0ns 0ns 0ns 10ns 20ns
va2 da2 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns
va3 da3 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns
va4 da4 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns

Vb1 db1 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb2 db2 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb3 db3 0 pulse 1.8 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs
Vb4 db4 0 pulse 1.80 0 0ns 1ns 1ns 10ns 20ns $ select inv inputs

.option scale=0.09u

M1000 dff_0/m1_2_51# dff_0/m1_0_n57# vdd dff_0/nand_1/w_n2_n3# CMOSP w=10 l=2
+  ad=100 pd=60 as=9850 ps=5960
M1001 dff_0/nand_1/a_n13_n30# clk gnd Gnd CMOSN w=5 l=2
+  ad=50 pd=40 as=4550 ps=3040
M1002 dff_0/m1_2_51# clk vdd dff_0/nand_1/w_n37_n3# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1003 dff_0/m1_2_51# dff_0/m1_0_n57# dff_0/nand_1/a_n13_n30# Gnd CMOSN w=5 l=2
+  ad=25 pd=20 as=0 ps=0
M1004 dff_0/m1_0_n57# dff_0/m1_2_51# vdd dff_0/nand_0/w_n2_n3# CMOSP w=10 l=2
+  ad=100 pd=60 as=0 ps=0
M1005 dff_0/nand_0/a_n13_n30# dff_0/m1_0_n126# gnd Gnd CMOSN w=5 l=2
+  ad=50 pd=40 as=0 ps=0
M1006 dff_0/m1_0_n57# dff_0/m1_0_n126# vdd dff_0/nand_0/w_n37_n3# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1007 dff_0/m1_0_n57# dff_0/m1_2_51# dff_0/nand_0/a_n13_n30# Gnd CMOSN w=5 l=2
+  ad=25 pd=20 as=0 ps=0
M1008 b1 dff_0/m1_166_52# vdd dff_0/nand_2/w_n2_n3# CMOSP w=10 l=2
+  ad=100 pd=60 as=0 ps=0
M1009 dff_0/nand_2/a_n13_n30# dff_0/m1_2_51# gnd Gnd CMOSN w=5 l=2
+  ad=50 pd=40 as=0 ps=0
M1010 b1 dff_0/m1_2_51# vdd dff_0/nand_2/w_n37_n3# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1011 b1 dff_0/m1_166_52# dff_0/nand_2/a_n13_n30# Gnd CMOSN w=5 l=2
+  ad=25 pd=20 as=0 ps=0
M1012 dff_0/m1_166_52# b1 vdd dff_0/nand_3/w_n2_n3# CMOSP w=10 l=2
+  ad=100 pd=60 as=0 ps=0
M1013 dff_0/nand_3/a_n13_n30# dff_0/m1_103_n118# gnd Gnd CMOSN w=5 l=2
+  ad=50 pd=40 as=0 ps=0
```



```
M1441 a_499_n884# a_307_n687# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1442 s3_b a_322_n849# vdd w_475_n857# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1443 s2_b s2 vdd w_214_n857# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1444 a3 a_422_836# a_448_817# Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1445 a_760_746# a_583_674# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1446 a_37_n706# a_11_n865# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1447 a2 a_126_836# vdd w_163_844# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1448 a_282_746# clk gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1449 a_n328_674# clk vdd w_n341_666# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1450 a_n292_n687# clk vdd w_n290_n857# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1451 a_422_836# a3 a_448_746# Gnd CMOSN w=5 l=2
+  ad=25 pd=20 as=0 ps=0
M1452 a_n14_615# a_n40_836# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1453 a_n40_658# a_n40_836# vdd w_128_666# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1454 a_619_n865# a_619_n687# vdd w_787_n857# CMOSP w=10 l=2
+  ad=0 pd=0 as=0 ps=0
M1455 a_645_n908# a_619_n687# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1456 a_37_n884# clk a_37_n908# Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
M1457 a_499_n706# a_307_n687# gnd Gnd CMOSN w=5 l=2
+  ad=0 pd=0 as=0 ps=0
C0 a1 gnd 0.28fF
C1 vdd dff_1/nand_2/w_n2_n3# 0.06fF
C2 a2 w_163_844# 0.04fF
C3 gnd s3_b 0.03fF
C4 cla_0/m1_85_n24# sum_0/c2 0.15fF
C5 w_n124_n857# a_n277_n849# 0.22fF
C6 a_307_n687# w_344_n857# 0.04fF
C7 vdd w_48_n679# 0.06fF
C8 m1_446_n329# cla_0/z55 0.05fF
C9 dff_3/nand3_0/a_n13_n54# dff_3/nand3_0/a_n13_n30# 0.09fF
C10 a_203_n777# gnd 0.08fF
C11 pg_0/w_338_50# pg_0/a_351_15# 0.04fF
C12 gnd db2 0.09fF
C13 dff_0/m1_103_n118# clk 0.09fF
C14 a_594_746# a_568_728# 0.19fF
C15 vdd a_n328_674# 0.97fF
C16 vdd w_n3_666# 0.12fF
C17 w_459_666# a_271_674# 0.27fF
C18 gnd a_203_n706# 0.08fF
C19 dff_1/m1_0_n126# dff_1/nand_3/w_n37_n3# 0.08fF
```

A total of 457 transistors are obtained in final extraction

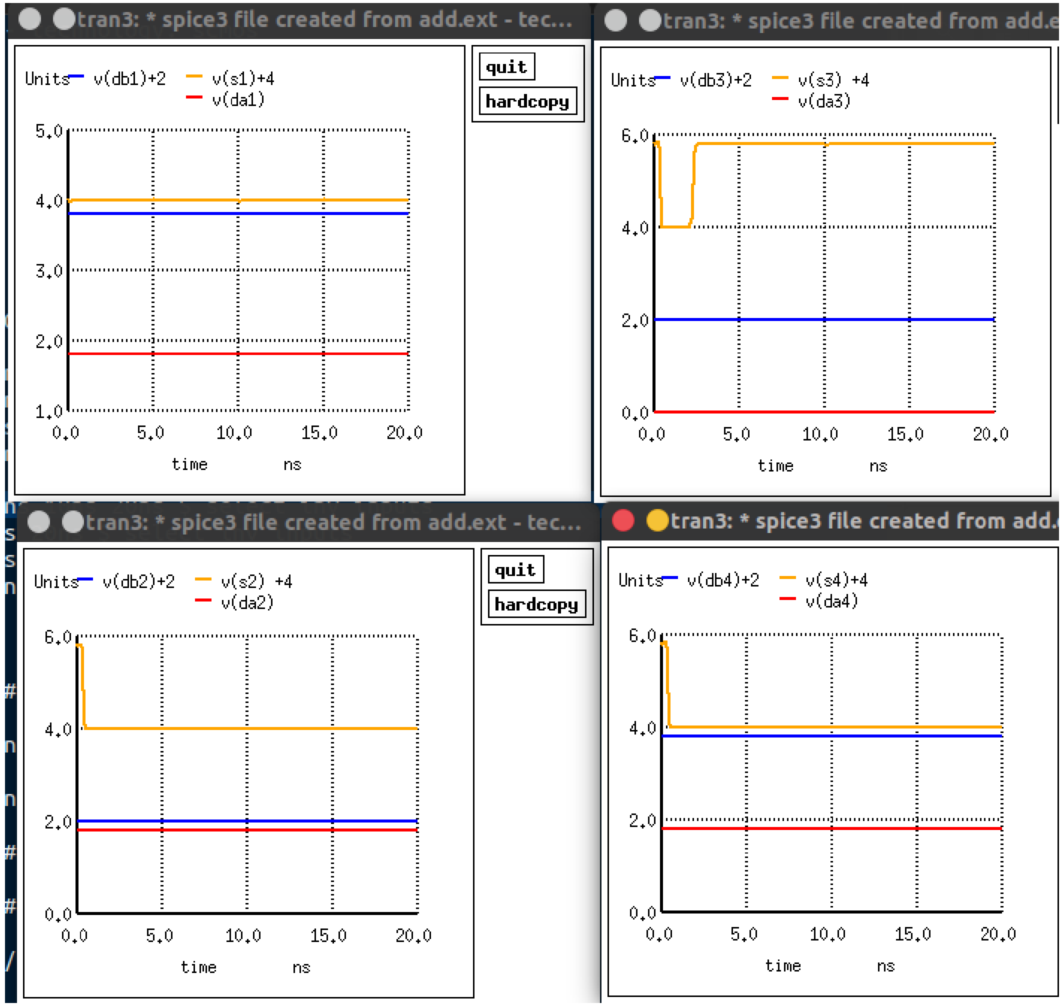
NETLIST

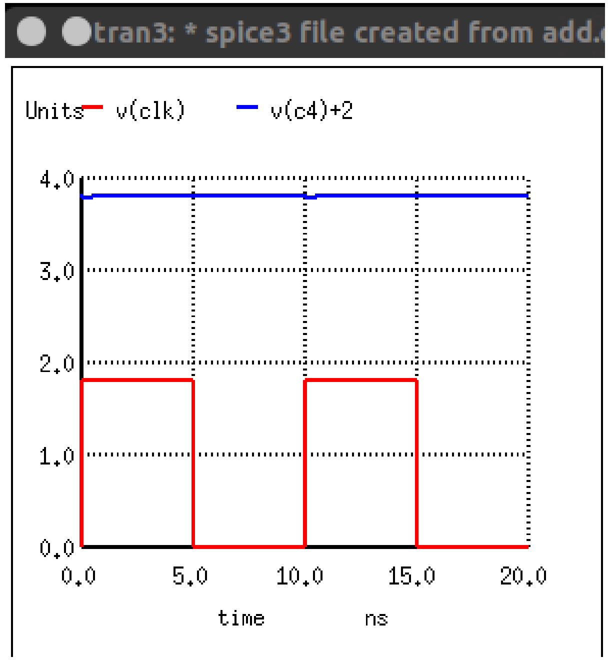


```
C2059 b2 Gnd 5.34fF
C2060 dff_1/m1_166_52# Gnd 1.09fF
C2061 dff_1/m1_2_51# Gnd 4.41fF
C2062 dff_1/nand_2/w_n2_n3# Gnd 0.70fF
C2063 dff_1/nand_2/w_n37_n3# Gnd 0.70fF
C2064 dff_1/nand_0/a_n13_n30# Gnd 0.08fF
C2065 dff_1/m1_0_n57# Gnd 2.53fF
C2066 dff_1/m1_0_n126# Gnd 3.03fF
C2067 dff_1/nand_0/w_n2_n3# Gnd 0.70fF
C2068 dff_1/nand_0/w_n37_n3# Gnd 0.70fF
C2069 dff_1/nand_1/a_n13_n30# Gnd 0.08fF
C2070 dff_0/nand3_0/a_n13_n54# Gnd 0.08fF
C2071 dff_0/nand3_0/a_n13_n30# Gnd 0.09fF
C2072 dff_0/nand3_0/w_33_n3# Gnd 0.70fF
C2073 dff_0/nand_1/w_n2_n3# Gnd 1.40fF
C2074 dff_0/nand_1/w_n37_n3# Gnd 1.40fF
C2075 dff_0/nand_4/a_n13_n30# Gnd 0.08fF
C2076 db1 Gnd 1.62fF
C2077 dff_0/nand_3/w_n2_n3# Gnd 1.40fF
C2078 dff_0/nand_3/w_n37_n3# Gnd 1.40fF
C2079 dff_0/nand_3/a_n13_n30# Gnd 0.08fF
C2080 dff_0/m1_103_n118# Gnd 0.98fF
C2081 dff_0/nand_2/a_n13_n30# Gnd 0.08fF
C2082 b1 Gnd 4.09fF
C2083 dff_0/m1_166_52# Gnd 1.09fF
C2084 dff_0/m1_2_51# Gnd 4.41fF
C2085 dff_0/nand_2/w_n2_n3# Gnd 0.70fF
C2086 dff_0/nand_2/w_n37_n3# Gnd 0.70fF
C2087 dff_0/nand_0/a_n13_n30# Gnd 0.08fF
C2088 dff_0/m1_0_n57# Gnd 2.53fF
C2089 dff_0/m1_0_n126# Gnd 3.03fF
C2090 dff_0/nand_0/w_n2_n3# Gnd 0.70fF
C2091 dff_0/nand_0/w_n37_n3# Gnd 0.70fF
C2092 dff_0/nand_1/a_n13_n30# Gnd 0.08fF



.tran 0.1n 20n

.control
set hcopypscolor = 1 *White background for saving plots
set color0=white ** color0 is used to set the background of the plot (manual sec:17.7)
set color1=black ** color1 is used to set the grid color of the plot (manual sec:17.7)


run
plot v(a1) v(b1)+2 v(ds1)+4
plot v(a2) v(b2)+2 v(ds2) +4
plot v(a3) v(b3)+2 v(ds3) +4
plot v(a4) v(b4)+2 v(ds4)+4
plot v(clk) v(c4)+2


.endc
```

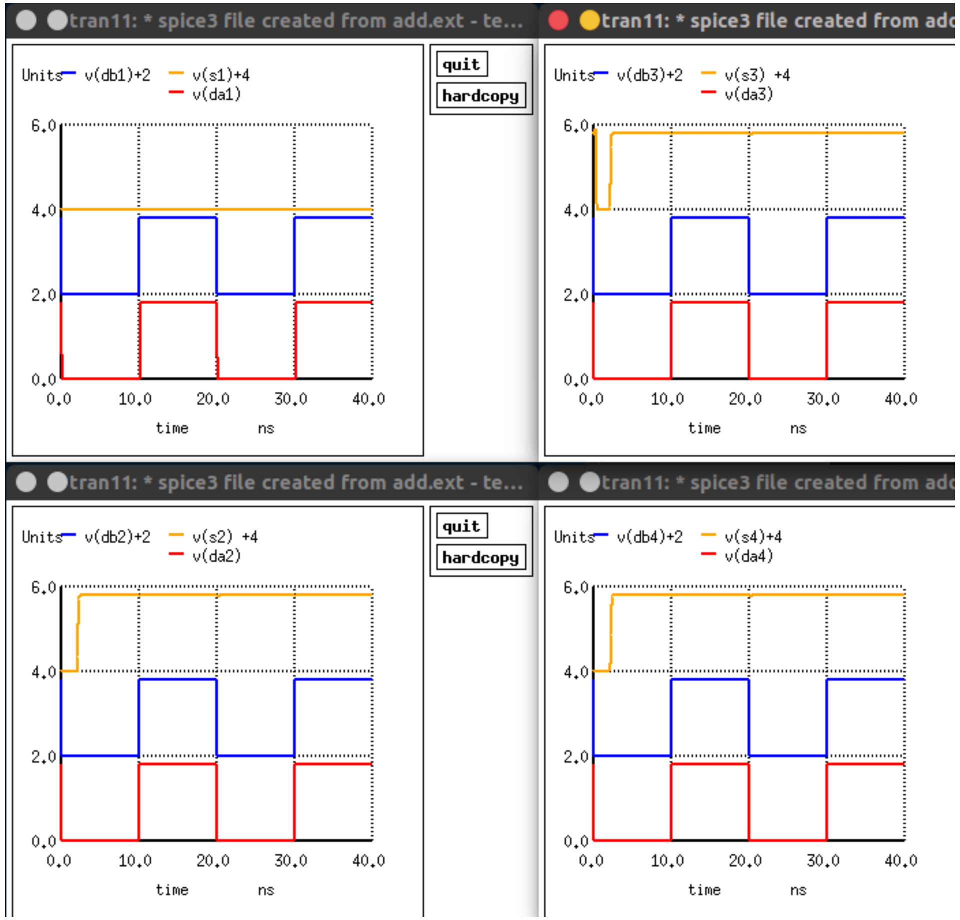
A total of 2092 capacitors are obtained in final extraction

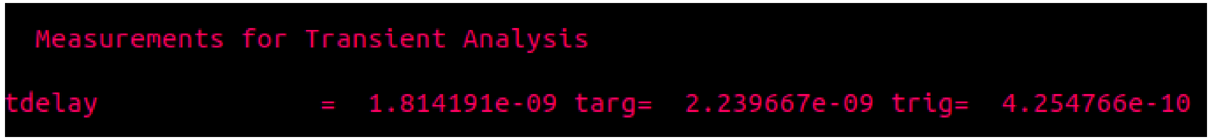
Plots


tran3: * spice3 file created from add.ext - tec...
quit
hardcopy
Units v(db1)+2 v(s1)+4 v(da1)
5.0 4.0 3.0 2.0 1.0
0.0 5.0 10.0 15.0 20.0
time ns
tran3: * spice3 file created from add.e
Units v(db3)+2 v(s3) +4 v(da3)
6.0 4.0 2.0 0.0
0.0 5.0 10.0 15.0 20.0
time ns
tran3: * spice3 file created from add.ext - tec...
quit
hardcopy
Units v(db2)+2 v(s2) +4 v(da2)
6.0 4.0 2.0 0.0
0.0 5.0 10.0 15.0 20.0
time ns
tran3: * spice3 file created from add.
Units v(db4)+2 v(s4)+4 v(da4)
6.0 4.0 2.0 0.0
0.0 5.0 10.0 15.0 20.0
time ns


tran3: * spice3 file created from add.
Units v(clk) v(c4)+2
4.0 3.0 2.0 1.0 0.0
0.0 5.0 10.0 15.0 20.0
time ns

Worst Case Delay





```
 Measurements for Transient Analysis

tdelay              =  1.814191e-09 targ=  2.239667e-09 trig=  4.254766e-10
```

Tclock >= TclkQ + Tdelay+ Tsetup. Here anyways Tdelay >> other quantities. So Tclock ~= Tdelay = 1.81nS

Fclock = 0.551GHz

Question 11

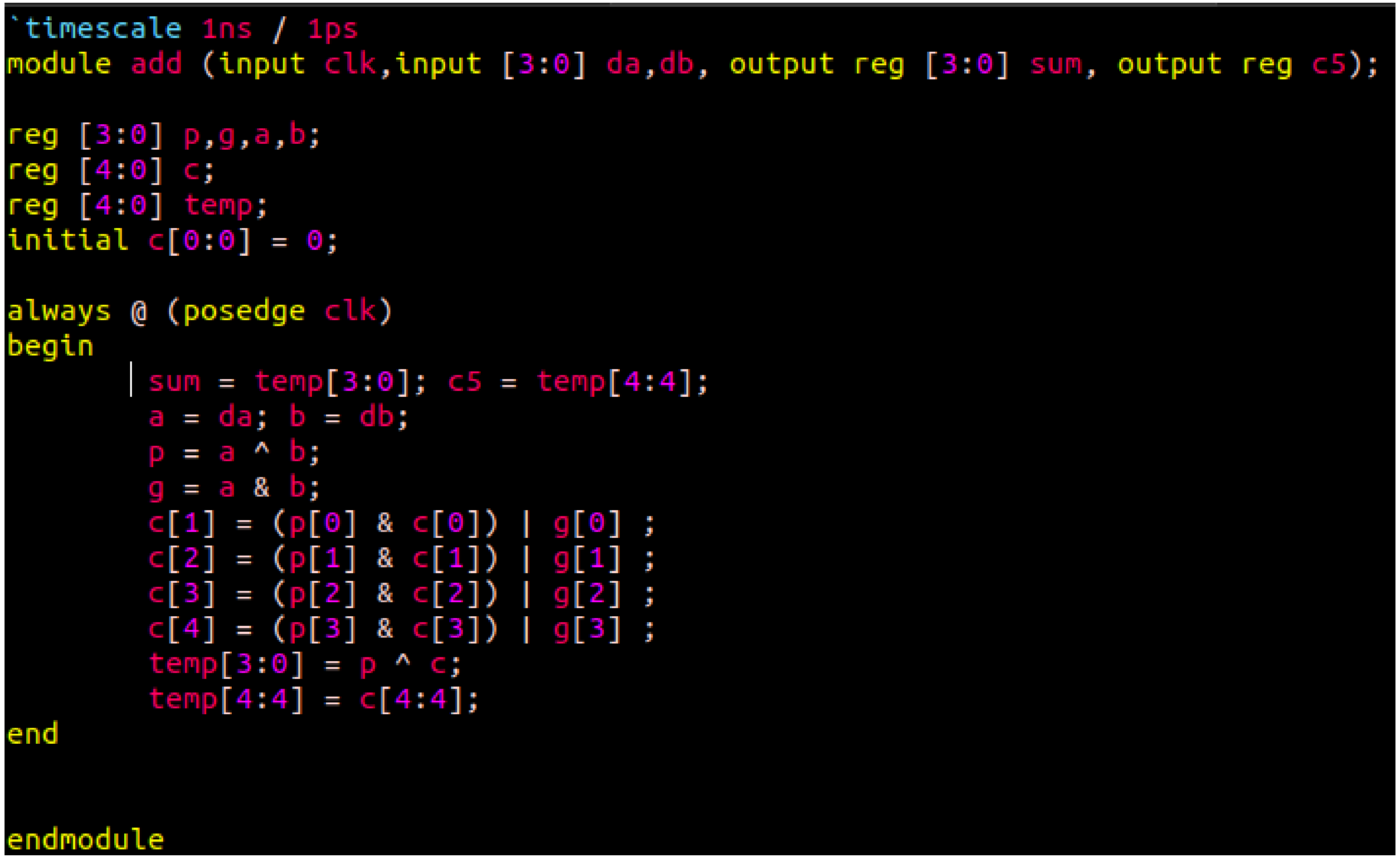
Using Verilog HDL write the structural description of your circuit and show the correctness of the functionality using simulations. Attach the required wave forms.

C

O

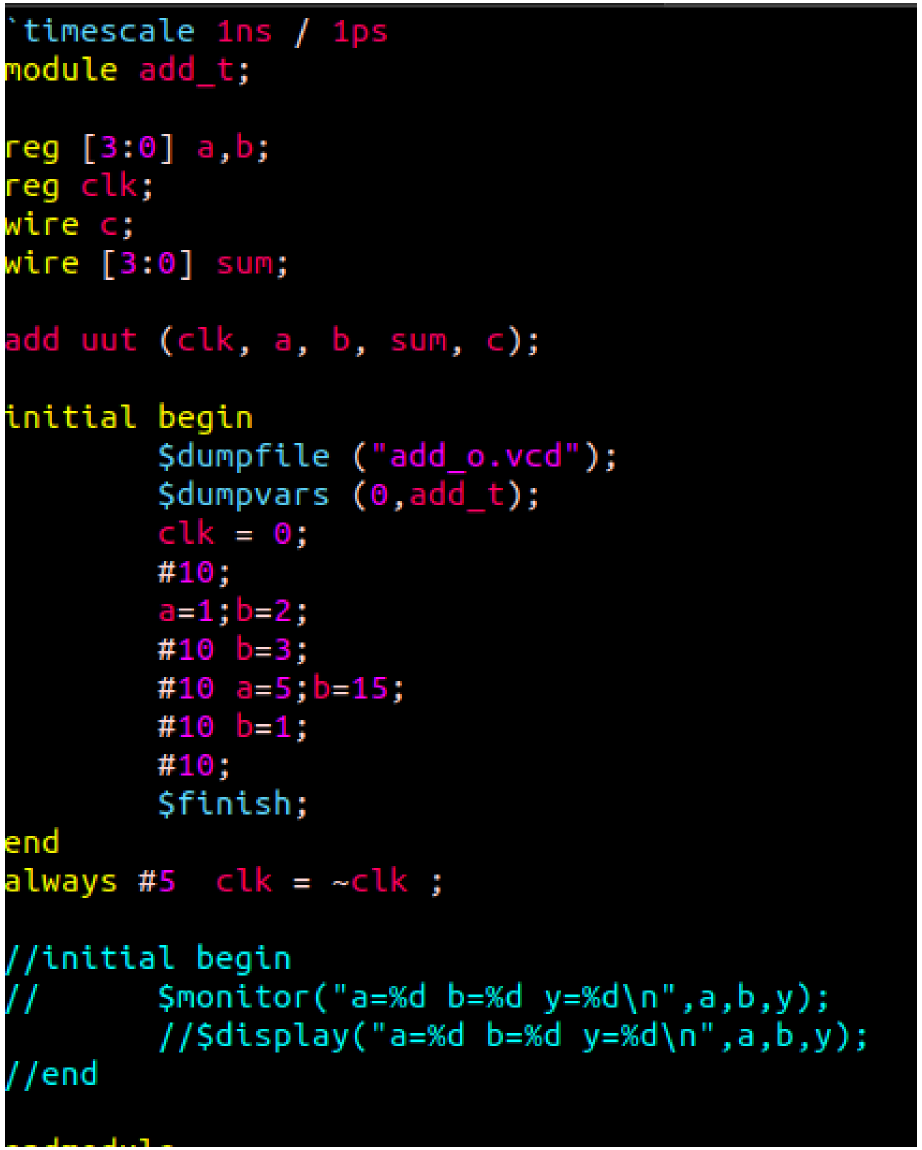
D

E



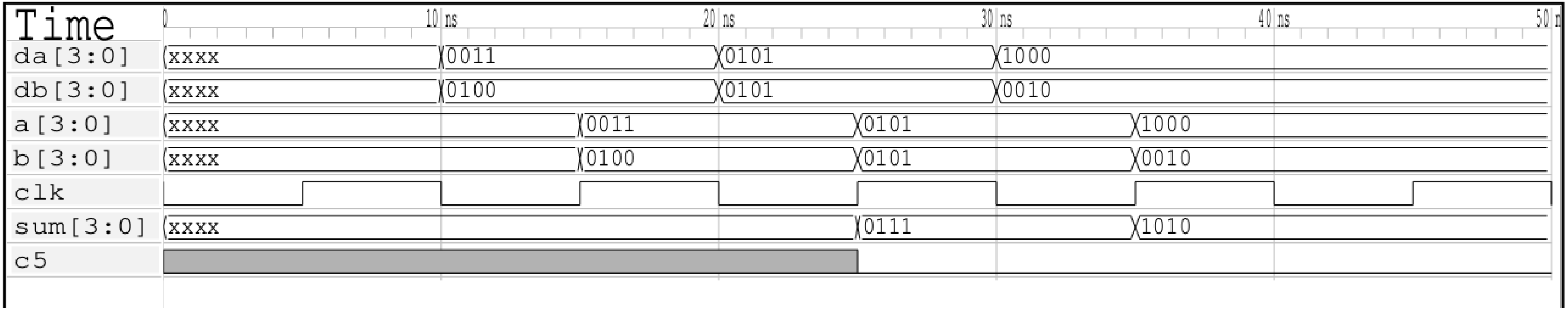
```
`timescale 1ns / 1ps
module add (input clk,input [3:0] da,db, output reg [3:0] sum, output reg c5);

reg [3:0] p,g,a,b;
reg [4:0] c;
reg [4:0] temp;
initial c[0:0] = 0;

always @ (posedge clk)
begin
        sum = temp[3:0]; c5 = temp[4:4];
        a = da; b = db;
        p = a ^ b;
        g = a & b;
        c[1] = (p[0] & c[0]) | g[0] ;
        c[2] = (p[1] & c[1]) | g[1] ;
        c[3] = (p[2] & c[2]) | g[2] ;
        c[4] = (p[3] & c[3]) | g[3] ;
        temp[3:0] = p ^ c;
        temp[4:4] = c[4:4];
end


endmodule
```

TESTBENCH



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

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

add uut (clk, a, b, sum, c);

initial begin
        $dumpfile ("add_o.vcd");
        $dumpvars (0,add_t);
        clk = 0;
        #10;
        a=1;b=2;
        #10 b=3;
        #10 a=5;b=15;
        #10 b=1;
        #10;
        $finish;
end
always #5  clk = ~clk ;

//initial begin
//      $monitor("a=%d b=%d y=%d\n",a,b,y);
        //$display("a=%d b=%d y=%d\n",a,b,y);
//end

endmodule
```

Plots


Time
0
10 ns
20 ns
30 ns
40 ns
50 n
da[3:0]
xxxx
0011
0101
1000
db[3:0]
xxxx
0100
0101
0010
a[3:0]
xxxx
0011
0101
1000
b[3:0]
xxxx
0100
0101
0010
clk
sum[3:0]
xxxx
0111
1010
c5

- In the structural description of the above netlist, where primitive gates are expected to be used, I've directly used the logical expressions as both finally perform the same operation without trading off any other parameter like time or anything.
- We can clearly observe that inputs are loaded into registers a and b on a positive edge while outputs are updated in the next positive edge clock.

Thank You