VLSI Assignment-3

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

Question 1

1. Size the pass transistor logic based multiplexor shown in Fig. 1 such that the average delay from input (A or B) to output (Y) is minimized. It is given that the electrical effort (C_{out}/C_{in}) for each path (input to output) is 2. Length of each transistor in the circuit is same (L=0.18 μm). Widths shown in the figure represents the ratio of PMOS and NMOS devices. There is a constraint for the output inverter that the total width $W_p + W_n = 6W$. Use NGSPICE simulations by including parasitic capacitances of the devices to find the sizes to minimize the delay. For your final design, show the functionality of the multiplexor with appropriate waveforms and report the delay of your circuit.

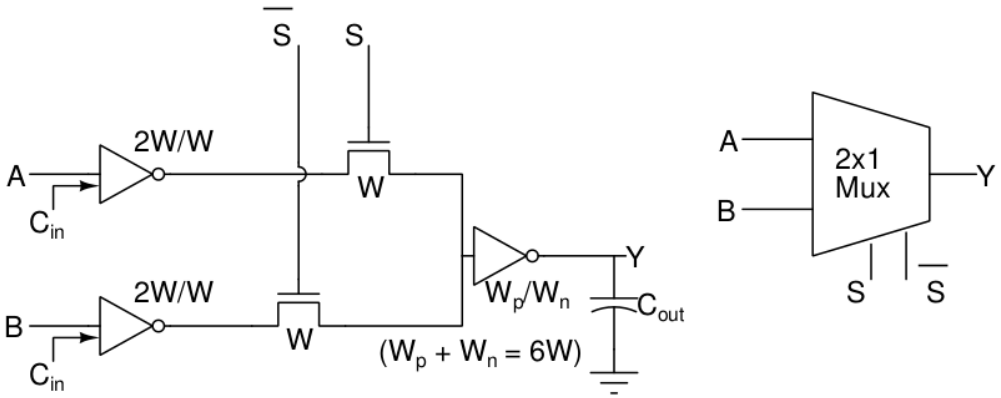


Figure 1

NETLIST

```
Viswanadh-2019112011
.include TSMC_180nm.txt
.param SUPPLY=1.8
.param LAMBDA=0.09u
.global gnd vdd
.param width_N1={5*LAMBDA}

Vdd     vdd     gnd     'SUPPLY'
vin a 0 pulse 0 1.8 0ns 1ns 1ns 10ns 25ns
vinb b 0 pulse 0 1.8 0ns 1ns 1ns 20ns 50ns
Vs s gnd pwl (0 0v 49.9ns 0v 50ns 1.8v 100ns 1.8v)
Vsb sb gnd pwl ( 0 1.8v 49.9ns 1.8v 50ns 0v 100ns 0v)


.subckt inv y x vdd gnd $ output, input, vdd, gnd
.param width_P={2*5*LAMBDA}
.param width_N={5*LAMBDA}
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 inv

.subckt inv1 y x vdd gnd $ output, input, vdd, gnd
.param width_P={0.21*5*LAMBDA} $ below this, there's breakdown
.param width_N={5.79*5*LAMBDA}
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 inv1

x1 c a vdd gnd inv $inverter
x2 d b vdd gnd inv $W-N = 20l
M11 c s e gnd CMOSN W={width_N1} L={2*LAMBDA} AS={5*width_N1*LAMBDA} $ NMOS
+ PS={10*LAMBDA+2*width_N1} AD={5*width_N1*LAMBDA} PD={10*LAMBDA+2*width_N1}
M12 d sb e 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}
x3 f e vdd gnd inv1 $ inverter1
```

```
C1 a gnd 2.16f  $ cap ~
C2 b gnd 2.16f
C3 e gnd 4.32f
C4 f gnd 4.32f

*.dc vin 0 1.8 0.1
.tran 0.1n 100n

** MEASURING DELAYS (Refer manual section 15.4.5)
.measure tran tpdrA
+ TRIG v(a) VAL='SUPPLY/2' RISE=1
+ TARG v(f) VAL='SUPPLY/2' RISE=1

.measure tran tpdfA
+ TRIG v(a) VAL='SUPPLY/2' FALL=1
+ TARG v(f) VAL='SUPPLY/2' FALL=1
.measure tran tpdA param='(tpdrA+tpdfA)/2' goal=0

.measure tran tpdrB
+ TRIG v(b) VAL='SUPPLY/2' RISE=1
+ TARG v(f) VAL='SUPPLY/2' RISE=1

.measure tran tpdfB
+ TRIG v(b) VAL='SUPPLY/2' FALL=1
+ TARG v(f) VAL='SUPPLY/2' FALL=1
.measure tran tpdB param='(tpdrB+tpdfB)/2' goal=0

.measure tran tpd param='(tpdA+tpdB)/2' goal=0

.control
set hcopypscolor = 1 *White background for saving plots
set curplottitle = "Viswanadh-2019112011"

run
*plot v(a)
*plot v(b)
*plot  v(s) plot v(sb)  plot v(a)  plot v(b) plot v(f)

.endc
```

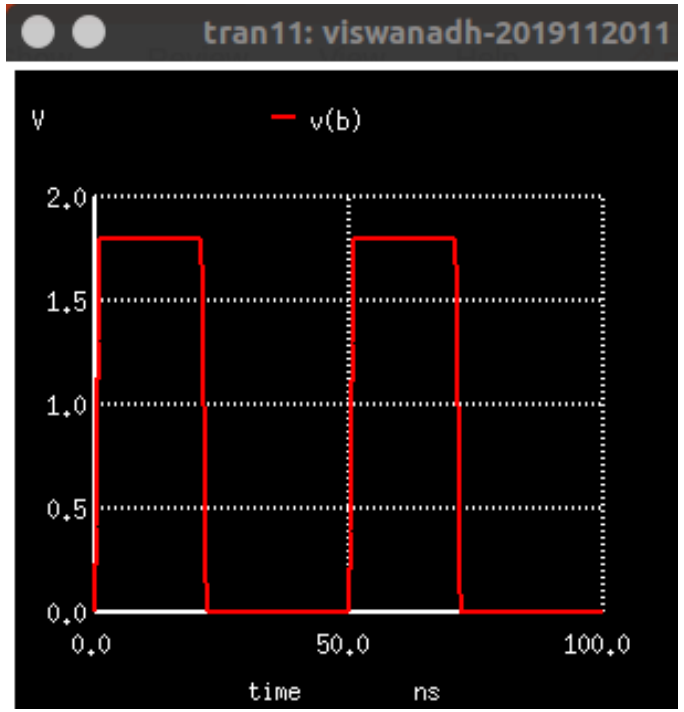
Plots

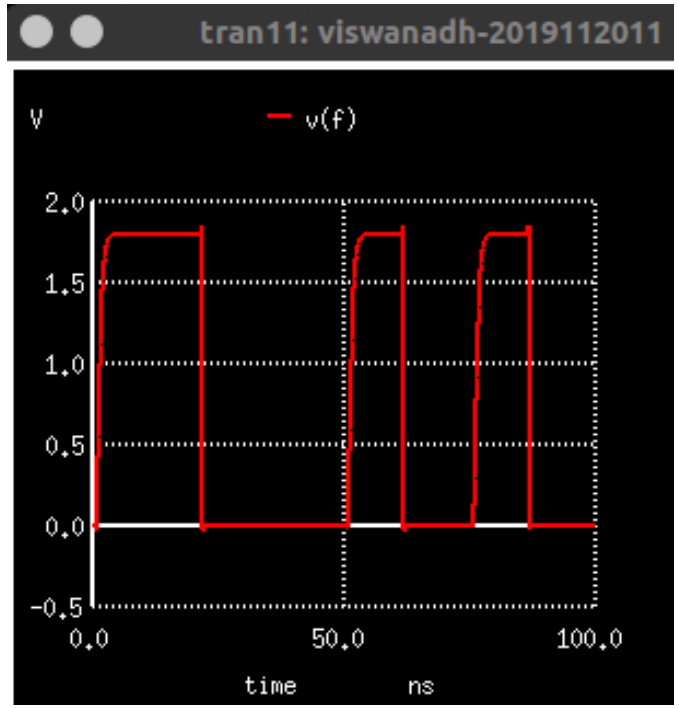
Plot of V_B , V_A , V_S and V_F vs time.

X-axis: time(in nS)

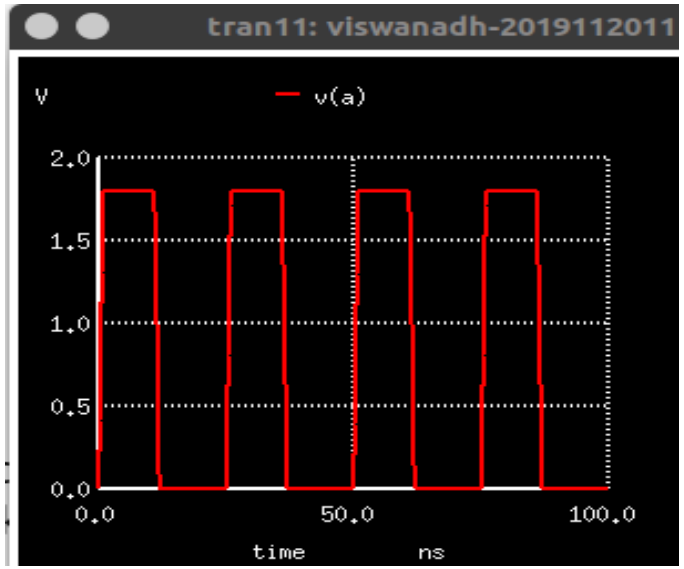
Y-axis: Voltage(in V)

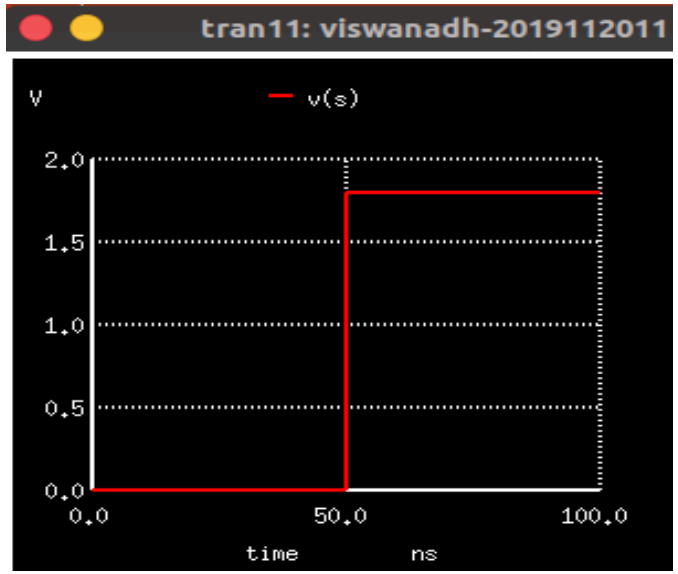
---- V





Here, A and B are inputs to the mux while s is the select line and finally, F is the output of the mux.





Finding Capacitances

From the given figure, C_{in} is equal to equivalent gate capacitance of the minimum-sized inverter

$\Rightarrow \quad C_G = WL \cdot C_{ox}$

we know $\quad C_{in} \propto C_G$

$\propto (W_p + W_N) L$

Here, $W_p = 2W$ & $W_N = W$

$\therefore \quad C_{in} \propto 3WL$

$\Rightarrow \quad C_{in} = 3WL\, C_{ox}$

Substituting the values,

$\lambda = 0.09\,\mu$

$L = 2\lambda$

$W = 5\lambda$ {considered}

$C_{ox} = 8.85 \times 10^{-3}$

now $C_{in} = 3(5\lambda)(2\lambda)(C_{ox})$

$= 3 \times 10 \times 81 \times 10^{-16} \times 8.85 \times 10^{-3}$

$= 2.15.. \times 10^{-15}\,C$

$\simeq 2.16\,fC$

now $C_{out} = 2 \cdot C_{in}$ {given}

$= 4.32\,fC$

C_{in} at the output inverter $= 4.16\,fC$

{as given $W_p + W_N = 6W$
$\propto 2 \cdot C_{in}$}

Explanations

- Mux is designed using subckt of inverters and are arranged as given in the question.
- For the two input ports A and B, oscillating square wave was given with different time periods (25,50ns). Select line toggles its state at t=50 ns hence for t<50, output is similar to plot of B as s=0 and for t>=50, output is similar to plot of A as s=1.
- To minimise the delay between path A to F and path B to F, we have the flexibilty to modify Wp/ Wn with a constraint between them ie Wp + Wn = 6W where W = 5* Lambda in my case with lambda = 0.09u.
- So for various combinations of Wp/Wn that are tried manually, the delay became minimum for Wp = 2.2 and Wn = 3.8 (with precision of 0.1). As we move away from these values, delays were increasing ultimately making the chosen combination the best one.
- Best values obtained were Wp = 2.2*5*Lambda and Wn = 3.8*5*Lambda

Wp	Wn	Avg delay*
1	5	2.84499e-09
2	4	2.82385e-09
3	3	2.82776e-09
4	2	2.85243e-09
5	1	2.94527e-09
2.2 (optimal)	3.8	2.82365e-09

* Here Avg delay = (TpA + TpB)/2 where TpA and TpB are delay periods when input A and B are considered respectively

```
tpdra              =  2.872379e-10 targ=  7.872379e-10 trig=  5.000000e-10
tpdfa              =  1.036006e-08 targ=  2.186006e-08 trig=  1.150000e-08
tpda               =  5.32365e-09
tpdrb              =  2.872379e-10 targ=  7.872379e-10 trig=  5.000000e-10
tpdfb              =  3.600575e-10 targ=  2.186006e-08 trig=  2.150000e-08
tpdb               =  3.23648e-10
tpd                =  2.82365e-09
```

Fig: Time Delays for optimal case

Question 2

2. Use the 2×1 multiplexor (only) shown in Fig. 2 and implement the following logic function *(Hint: Use Shanon's expansion.)*:

$$f = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$

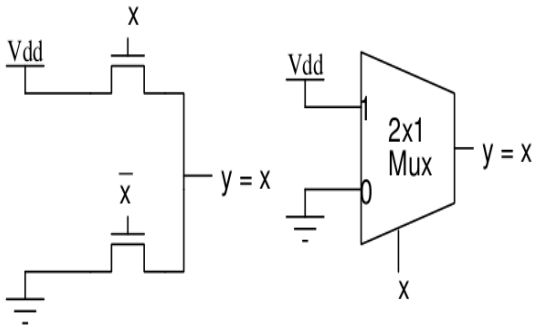


Figure 2

(a) Give the circuit diagram

(b) Write the spice netlist for the circuit and verify the functionality with simulations. Attach your plots.

(c) Find the minimum transition time taken by output (f) to change from - i) 0 to 1 (t_{PLH}) and ii) 1 to 0 t_{PHL}. Clearly mention the input combination and paths for charging and discharging the output in your circuit.

(d) Do you observe any difference in t_{PLH} and t_{PHL}. If yes, then can you suggest some modification in your circuit to make them nearly equal. *(Hint: Charging path (V_{DD} to out) and discharging path (out to ground) should have equal number of transistors of same size. Repeaters (inverters) can be inserted at appropriate nodes to achieve the same.)*

Each input (x_1 to x_4) should see a high input impedance (goes to gates of pass transistors) and the load capacitance at the output is equivalent to 4 times the minimum sized inverter. Use the same size (W) for pass transistor obtained in the previous problem.

Part A)

Given

$$f = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$

From Shannon's expansion,

$$f = x_1\underbrace{(x_2 + x_3 + x_4 + x_2x_3 + x_2x_4 + x_3x_4)}_{B} + \bar{x}_1\underbrace{(x_2x_3 + x_2x_4 + x_3x_4)}_{A}$$

$$A = x_2x_3 + x_2x_4 + x_3x_4$$

$$= x_2(x_3 + x_4 + x_3x_4) + \bar{x}_2(x_3x_4)$$

$$= x_2\underbrace{(x_3 + x_4)}_{C} + \bar{x}_2\underbrace{(x_3x_4)}_{D}$$

$$C = x_3(1) + \bar{x}_3(x_4)$$

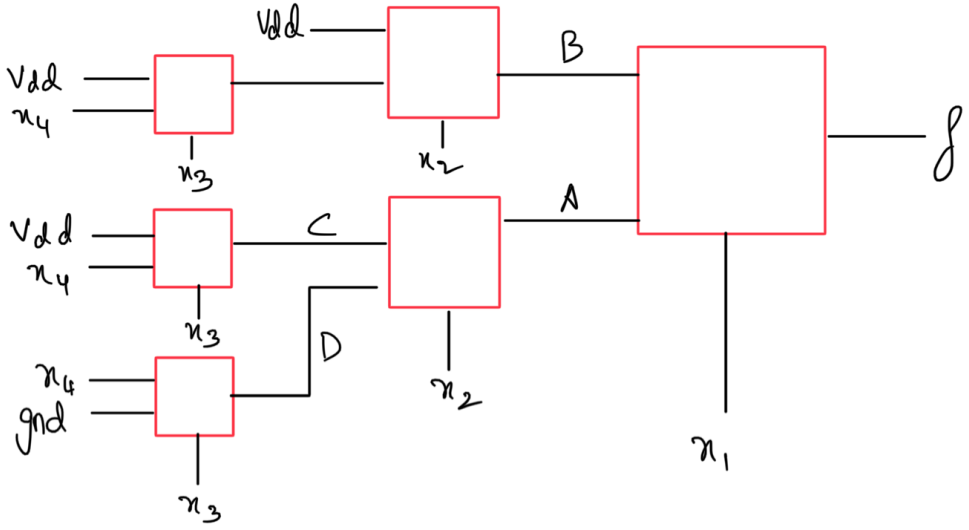
$$D = x_3(x_4) + \bar{x}_3(0)$$

B: (simplified exp) $= x_2 + x_3 + x_4$

$$= \bar{x}_2(x_3 + x_4) + x_2(1)$$

$$= x_2(1) + \bar{x}_2\left(x_3(1) + \bar{x}_3(x_4)\right)$$

* Representing mux with squares,



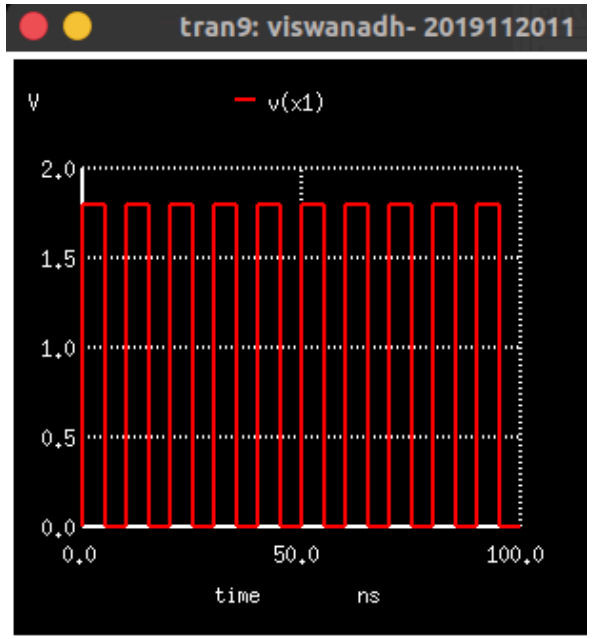
Part B)

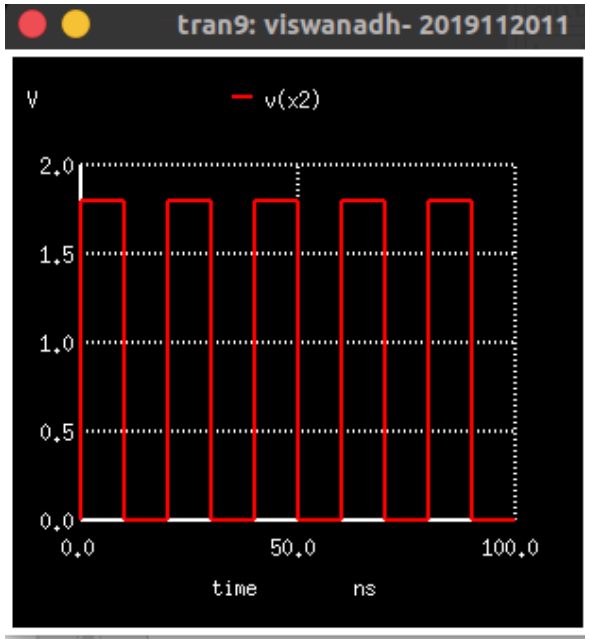
NETLIST

```
Viswanadh- 2019112011
.include TSMC_180nm.txt
.param SUPPLY=1.8
.param LAMBDA=0.09u
.param width_N={5*LAMBDA}
.global gnd vdd

Vdd vdd gnd 1.8
Va x1 gnd pulse 0 1.8 0ns 100ps 100ps 5ns 10ns $select inputs
Vb x2 gnd pulse 0 1.8 0ns 100ps 100ps 10ns 20ns
Vc x3 gnd pulse 0 1.8 0ns 100ps 100ps 20ns 40ns
Vd x4 gnd pulse 0 1.8 0ns 100ps 100ps 40ns 80ns

Ve x1b gnd pulse 1.8 0 0ns 100ps 100ps 5ns 10ns $ select inv inputs
Vf x2b gnd pulse 1.8 0 0ns 100ps 100ps 10ns 20ns
Vg x3b gnd pulse 1.8 0 0ns 100ps 100ps 20ns 40ns
Vh x4b gnd pulse 1.8 0 0ns 100ps 100ps 40ns 80ns

.subckt mux i1 i2 y x xb gnd $ inputs output select and inv vvdd gnd
M1 i1 x 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 i2 xb 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 mux

Xz vdd gnd d x4 x4b gnd mux $ creating 4 required input mux points
```

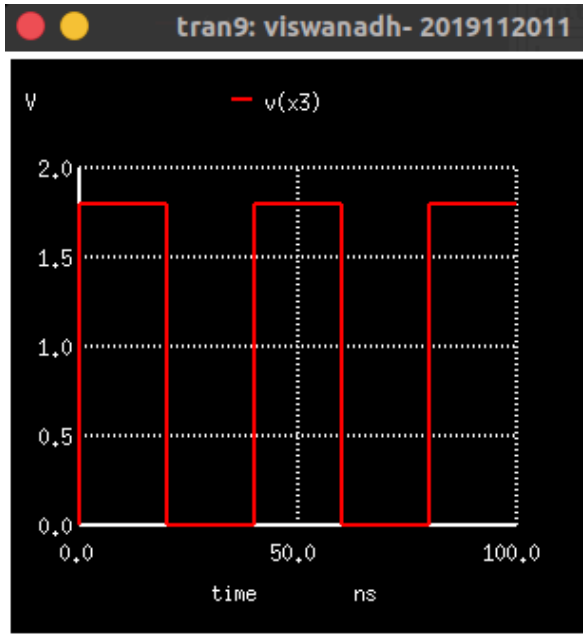
```
Xa d gnd o4 x3 x3b gnd mux $ third level
Xb vdd d o5 x3 x3b gnd mux
Xf vdd d o6 x3 x3b gnd mux

Xc o5 o4 o2 x2 x2b gnd mux $ second level
Xd o7 o6 o3 x2 x2b gnd mux

Xe o3 o2 o1 x1 x1b gnd mux $ first level

Cout o1 gnd 8.64f

*.dc vin 0 1.8 0.1
.tran 0.1n 100n

** MEASURING DELAYS (Refer manual section 15.4.5)
.measure tran tplh
+ TRIG v(o1) VAL='0' RISE=1
+ TARG v(o1) VAL='SUPPLY*0.5' RISE=1
.measure tran tphl
+ TRIG v(o1) VAL='SUPPLY*0.5' FALL=1
+ TARG v(o1) VAL='0' FALL=1

.control
set hcopypscolor = 1 *White background for saving plots
set curplottitle = "Viswanadh-2019112011"

run
plot  v(o1)
plot  v(x1)
plot  v(x3)
plot v(x2)
plot v(x4)

.endc
```

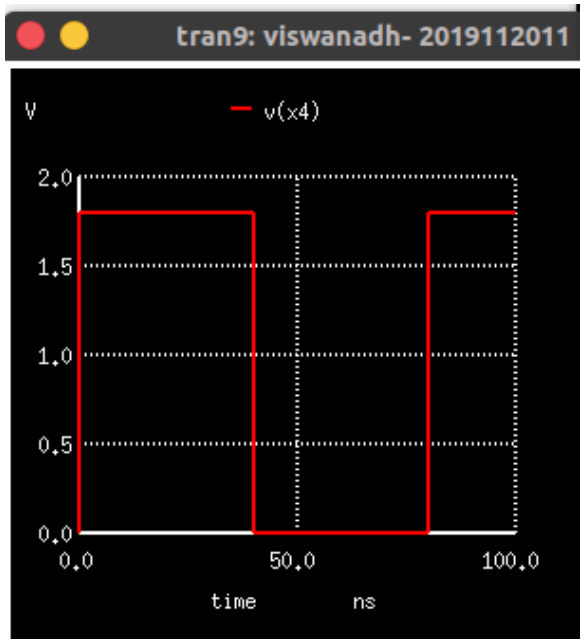
Plots

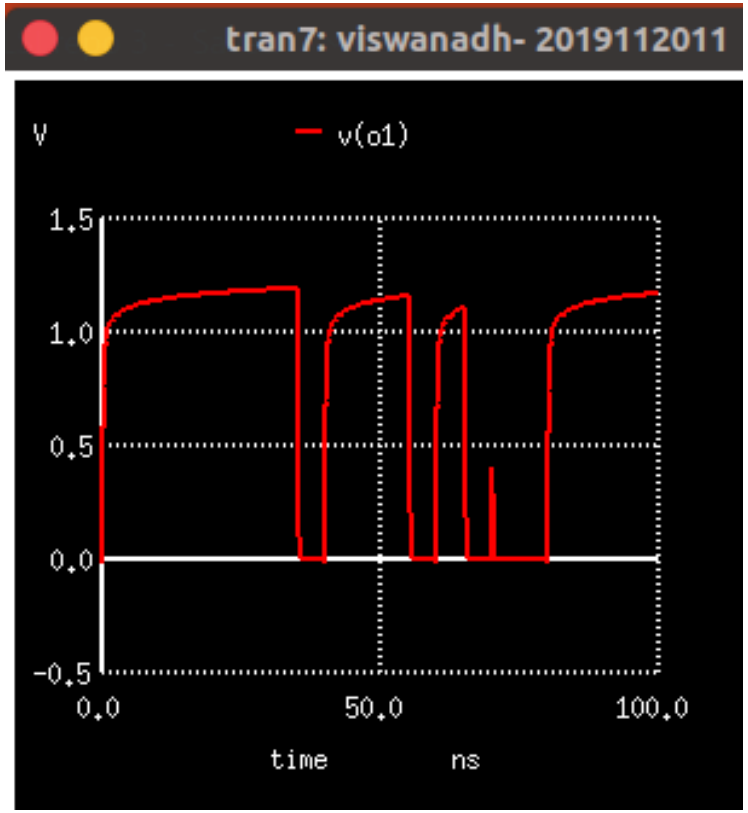




INPUTS







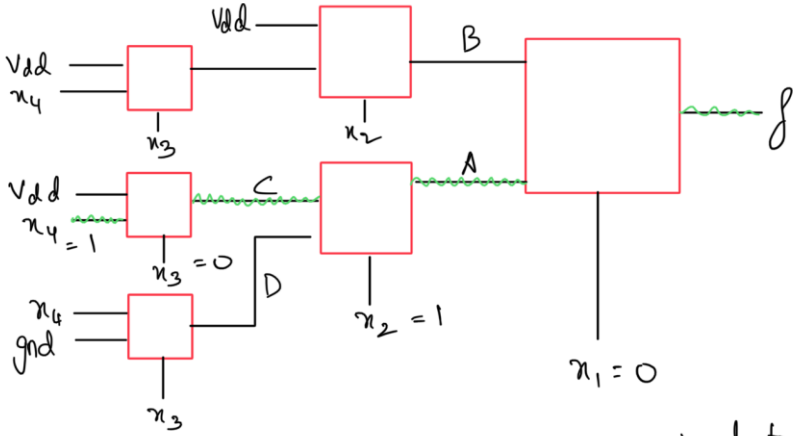
OUTPUT

- In the netlist above, firstly a subckt was created to replicate a MUX module.
- Then this subckt was used repeatedly and arranged accordingly as shown in the subpart A).
- In previous question, we obtained C_{in} = 2.16 fC. So now C_{out} is 4*C_{in} as it was mentioned output capacitance is 4 times the inverter (min). Now various inputs were given to x_1,x_2,x_3,x_4 and their corresponding outputs are observed from port f, i.e. V(f) is plotted.
- The outputs correctly match the theoretical values which were calculated from the expression given in the question.

Part C)

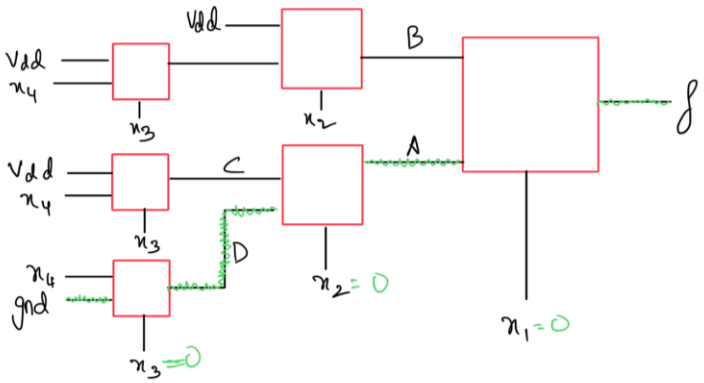
time \ value	x_1	x_2	x_3	x_4	f	
t = 0 nS	1	1	1	1	1	→ minimum delay charging path
t = 25 nS	0	1	0	1	1	→ Charging path
t = 75 nS	0	0	0	0	0	→ Discharging path

For input combination ①,
{green line is the charging path}



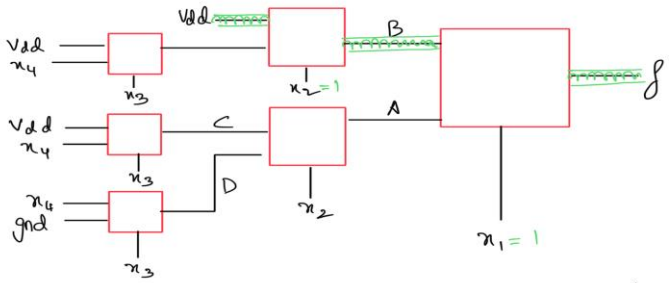
Here the delay caused would be proportional to delay caused by sum of 3 NMOS devices.

(ii) For input combination ②, {green line is the discharging path}



Here again, delay would be proportional to delay caused by sum of 3 NMOS devices

When $x_1 = 1$; $x_2 = 1$; x_3 & x_4 can be anything we can observe this path will have lowest delay



It's intuitive to figure out that this path has lowest delay as the path has only 2 muxes (Transistors). Also this is a path for charging

- It can be noted that we have 2 types of paths for charging, where one path encounters only 2 MUXes (transistors) while another path has 3 MUXes.
- Charging with minimum delay occurs at t=0 sec and hence for calculating delay, I've considered RISE =1 as the circuit will also start at the same time (I.e 0 nS)
- But for discharging, it can be noted that all paths have 3 transistors in their paths and hence delays caused would be more or less the same. So finding delay for any value of RISE gives us the similar values

- I’ve added a new MUX to make transistor count same for all charging/ discharging paths.
- But we were told to use an inverter which has 2 active transistors when it’s included in a path.
- But I need only a single transistor to make the count same, hence i’ve used a MUX instead of a inverter as MUX has only 1 active transistor.
- * *Active transistor* phrase is used only to imply that all transistor needn’t be included in the count for a charging/ discharging path.

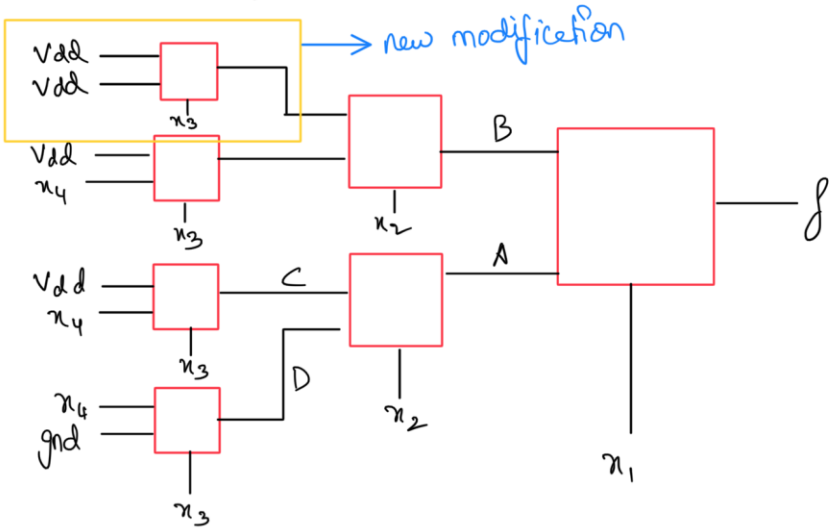
```
 Reference value :   1.76433e-09
No. of Data Rows : 1165

  Measurements for Transient Analysis

tplh                 =  3.844610e-10 targ=  4.316905e-10 trig=  4.722955e-11
tphl                 =  7.772477e-10 targ=  3.597827e-08 trig=  3.520102e-08
```

	T_{PLH} (in S)	T_{PHL} (in S)
Minimum delays	3.844610e-10	7.772477e-10

- The input combination and their paths for charging/ discharging is shown in the previous slide.
- Now, minimum transition times T_{PLH} and T_{PHL} for those paths were also found using the TARG and TRIG functions.
- Here T_{PLH} is the time taken for the circuit to charge its output port from 0 to 1 (HIGH) while T_{PHL} is the time taken for the circuit to discharge its output port from 1 to 0 (LOW).

Part D)

We can observe difference in values of t_{pdn} & t_{pdr}
This can be optimised by making transistors equal
in all charging / discharging paths. For this,
from the circuit it is evident that all paths
pass through 3MUX units except for the path
on the top which passes through only 2MUX modules
Hence then, we would replace VDD with another
MUX unit whose both inputs are VDD & select line
can be any one of x_1, x_2, x_3 or x_4.




```
Xz vdd gnd d x4 x4b gnd mux $ creating 4 required input mux points

Xh vdd vdd o7 x3 x3b gnd mux
Xa d gnd o4 x3 x3b gnd mux $ third level
Xb vdd d o5 x3 x3b gnd mux
Xf vdd d o6 x3 x3b gnd mux

Xc o5 o4 o2 x2 x2b gnd mux $ second level
Xd o7 o6 o3 x2 x2b gnd mux

Xe o3 o2 o1 x1 x1b gnd mux $ first level
```

* modified netlist including the new modification

```
 Measurements for Transient Analysis

tplh                 =  5.941726e-10 targ=  6.421113e-10 trig=  4.793869e-11
tphl                 =  7.773431e-10 targ=  3.597804e-08 trig=  3.520070e-08
```

(optimized circuit)	T_{PLH} (in S)	T_{PHL} (in S)
Minimum delay	5.941726e-10	7.773431e-10

- Now, minimum transition times T_{PLH} and T_{PHL} for those paths were also found using the TARG and TRIG functions.
- Here T_{PLH} is the time taken for the circuit to charge its output port from 0 to 1 (HIGH) while T_{PHL} is the time taken for the circuit to discharge its output port from 1 to 0 (LOW).
- It can be observed that T_{PLH} and T_{PHL} are almost near to each than in the previous case and now this can be attributed to the fact all the charging/ discharging paths have same number of transistors (MUXes) that result in same/ similar delays.
- Here T_{PLH} is similar to T_{PHL}

Question 3)

3. *(GCD control unit design, Reference - Computer Architecture and Organisation, third edition by John P Hayes):* You are asked to design control unit of a simple digital circuit that can find the greatest common divisor (GCD) of two given positive integers. A variant of Euclid's Algorithm proposed by Cormen, Leiserson and Rivest in 1990 is given in Fig. 3, which can be used to find the GCD of two integers X and Y. For example GCD(20,16)=4 and GCD(27,17)=1.

   From the algorithm shown in Fig. 3, it is identified that following sub-circuits will be required to implement the complete circuit- 1) registers to hold X and Y data, 2) comparator unit ($X > 0$ and $X \leq Y$), 3) subtracter unit (X = X - Y) and 4) set of multiplexers to realize the swap operation. The input to the circuit will be X, Y, a clock signal (CLK) and a reset signal. The circuit will also need control signals (instructions) such as subtract, swap, load-X and load-Y, etc.. Fig. 4 depicts the block diagram to realize the GCD algorithm by showing the data-path unit and control unit of the circuit.

   In order to generate control signals or instructions, we need to evolve a finite state machine (FSM) for the control unit shown in Fig. 4. The 4 states shown in Fig. 5 can be used to build the FSM of the circuit.

```
gcd(in: X,Y; out: Z);
  register XR, YR, TEMPR;
  XR := X;                          {Input the data}
  YR := Y;
  while XR > 0 do begin
    if XR ≤ YR then begin           {Swap XR and YR}
      TEMPR := YR;
      YR := XR;
      XR := TEMPR; end
      XR := XR – YR;                {Subtract YR from XR}
  end
  Z := YR;                          {Output the result}
end gcd;
```

Figure 3

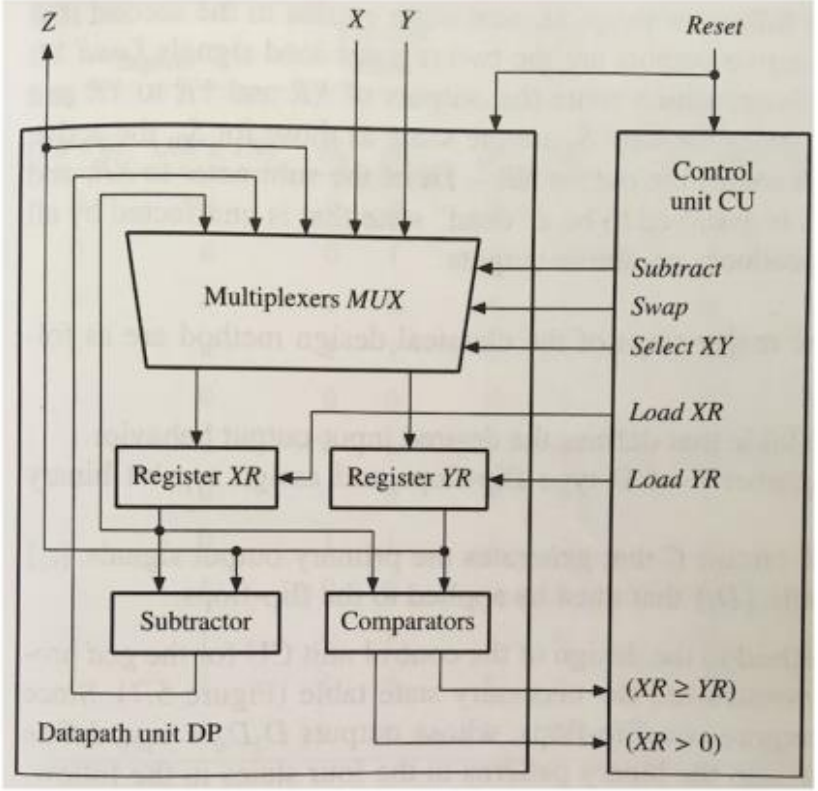


Figure 4

Part A)

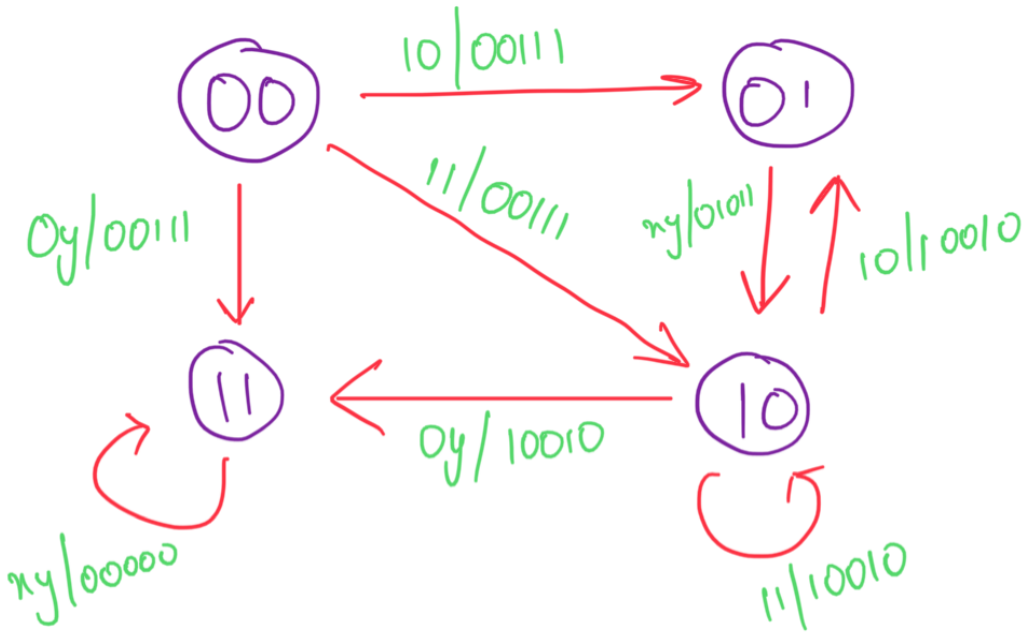
With the help of above informations, do the following:

(a) Build the state diagram corresponding to the state table shown in Fig. 5.

Rewriting the State Table,

Present State (A,B)	Next State: In: (x,y) 00	01	10	11 A(t+1) B(t+1)	Output {depends only present state} Z_1	Z_2	Z_3	Z_4	Z_5
00	11	11	01	10	0	0	1	1	1
01	10	10	10	10	0	1	0	1	1
10	11	11	01	10	1	0	0	1	0
11	11	11	11	11	0	0	0	0	0

where $x = XR > 0$ & $y = XR \geq YR$



* Don't care condition is referred with variables x or y here.

- From the given table in the question, states S_0,..,S_3 are replaced with binary representation of 00,..,11.
- 5 outputs of the control unit are labelled as z_1,..,z_5 (sub, swap,...)
- Control variables XR>0 and XR>=YR are labelled as x and y.
- A simplified state table with above notations is drawn in the previous slide and accordingly state diagram is also drawn with the four states representing S_0,..,S_3 respectively.
- In the state table, numbers on the arrows are of the form $xy/z_1z_2z_3z_4z_5$.

Part B)

Give the circuit diagram of the control unit using digital blocks (gates/muxes/flip-flops).

Circuit Diagram of CU

From the previous diagram, it's apparent that we would be using 2 flip-flops (D) – one for A other B

now as considered before, let the flipflops be D_A & D_B

AB \ xy	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	0

$A(t+1) = x' + y + B$

Similarly, for $B(t+1)$,

D_A & D_B

AB \ xy	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	1	1	1	1
10	1	1	0	1

For $B(t+1) =$
$= AB + Bx' + y'B'$

Now we know that the 5 outputs only depend on A & B

From the truth table,

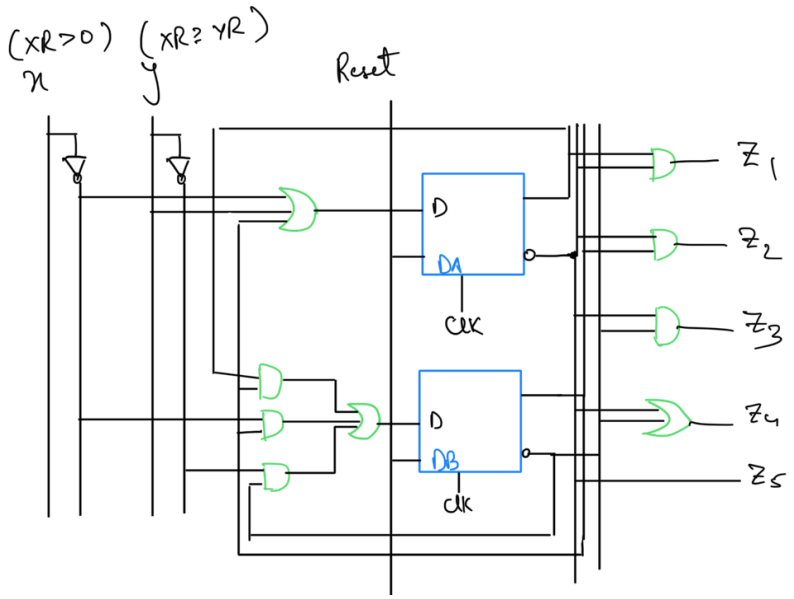
$Z_1 = AB'$

$Z_2 = A'B$

$Z_3 = A'B'$

$Z_4 = A' + B'$

$Z_5 = A'$



- I've chosen to build the circuit using D flipflops and number of flip flops required would be $\log_2 4 = 2$. Let these be D_A and D_B
- Now relation is framed between $D_A(t+1)$ and $D_B(t+1)$ in terms of present values x, y, $D_A(t)$ and $D_B(t)$. This involved constructing 4 variable Kmaps for the 4 variables mentioned and such 2 such Kmaps are constructed one each for $D_A(t+1)$ and $D_B(t+1)$
- Then the final required 5 outputs $z_1,..,z_5$ are constructed in terms of A and B as those values are obtained solely from the flipflops.
- Then from the derived expressions, circuit diagram is drawn.

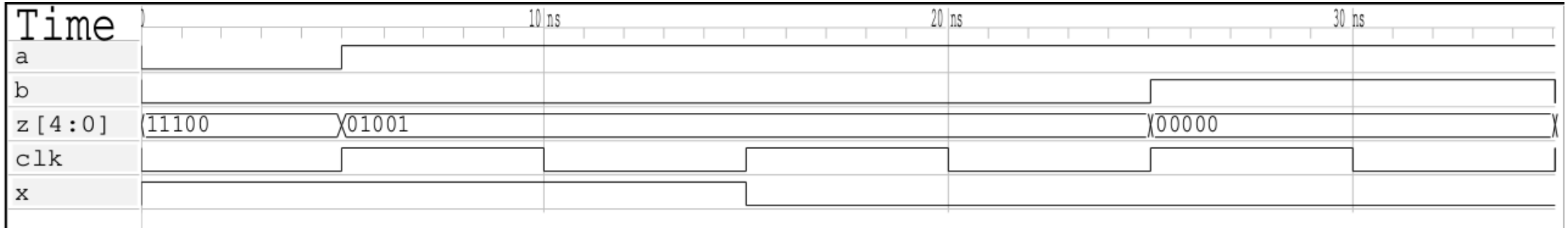
Part C)

(c) Implement the control unit circuit using Verilog HDL and show its functionality with simulation results. Use positive edge clock transitions in your design.

MODULE

```
`timescale 1ns / 1ps
module cu (input x,y, reset,clk, output reg [4:0] z,output reg a,b,c);//x: xr>0 y: xr>=yr
reg t;
initial begin
        a=0;b=0;c=1;
        z = 5'b11100;
end

always @ (posedge clk, reset)
begin
//      $display("second a=%b b=%b z=%b x=%b y=%b\n",a,b,z,x,y);
if (reset==1)
begin
$display("Reset trig\n");
a=0;b=0;c=!c; z = 5'b11100;
end
 t=b;//temp
 b = a*b + b*(!x) + (!y)*(!b);
 //a = !x + y + t; $original exp
 a = !(x* (!y)*(!t));
 z[0:0] = a*(!b);
 z[1:1] = b*(!a);
 z[2:2] = (!a)*(!b);
 z[3:3] = !(a*b);
 z[4:4] = (!a);


//      $display("sec done a=%b b=%b z=%b\n",a,b,z);
        c=!c;
 end
endmodule
```

- In the Verilog code attached, most of the expressions are similar to those derived in part B). It can be noted that gates are used to evaluate the expression and hence this is **not** a behavioral description.
- Here we are maintaining 2 variables a, b to denote the present state of the system and accordingly the output bits are set. Also, an additional variable c is toggled every time to trigger the datapath module.
- All the five output variables are stored in a vector z which is again passed to the datapath module.



Here, I tried out GCD (20,10). We can observe that initially (a,b) are set to 0,0. Then in the next clock cycle, a is toggled to 1 and correspondingly the output bits z[4:0] are set to subtraction stage (S_2). This subtraction continues until one of the numbers stored in registers becomes 0 where the stage is shifted to S_3 (end) , (a,b) becomes (1,1) which occurs at the 25th nS

Part D)

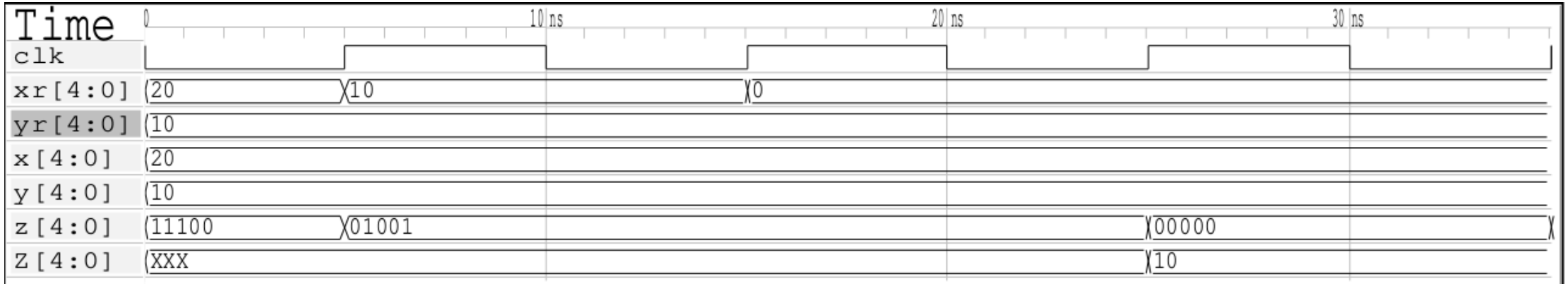
(d) Give a behavioural description of the data path unit using Verilog HDL and show the complete functionality of your circuit with simulation results

MODULE

```
`timescale 1ns / 1ps
module dp (input [4:0] z,x,y, output reg [4:0] xr,yr,tempr, input reset,clk,c, output reg [4:0
] zr,Z,input a,b);
reg t;
always @ (c,reset)
begin
//      $display("one z=%b\n",z);
        if(a==1 && b==1) begin Z = zr;$display("Z=%d\n",Z);#10; $finish;end
        if (z[2:2] == 1) // load
        begin
                if (z[3:3] == 1)
                xr=x;
                if (z[4:4] == 1)
                yr=y;
        end
        if (z[1:1] == 1) //swap
        begin
        tempr = yr;
        yr = xr;
        xr = tempr;
        end
        if (z[0:0] == 1) xr = xr- yr; //subtractor
        zr = yr;
//      $display("one done xr=%d yr=%d\n",xr,yr);
end

endmodule
```

- In the HDL code attached, most of the expressions are similar to those given in figure 3). It can be noted that if.. else conditions are used to evaluate the expression and hence this is a behavioral description.
- Here vector z is set in the control unit module and accordingly changes are made in this module.
- These changes include subtraction, swapping, loading values into registers.
- Also the present state values are used to identify the terminating condition of the code and that occurs when a=1 and b=1 (S_3).

PLOTS



- Here, I tried out GCD (20,10). We can observe that initially input values (stored in x and y) are loaded into registers xr and yr.
- Subsequently on the basis of output signals from control unit, the values in the registers are correspondingly swapped or subtracted.
- Once all the terminating condition is triggered , final answer is loaded in the register Z and in this case GCD (20,10) = 10

Part E)

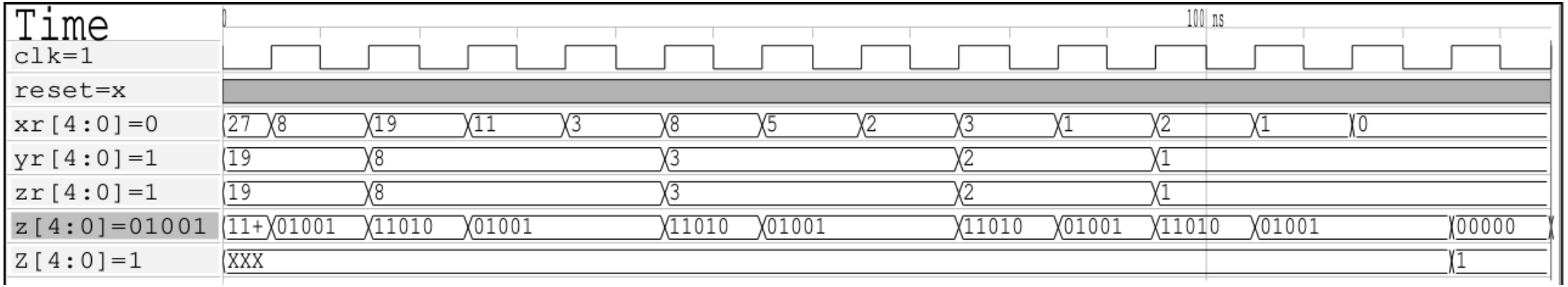
(e) Compute GCD of (27,19) and (24,16) with the help of your HDL model and show the simulation results

NETLIST

```
`timescale 1ns / 1ps
`include "dp.v"
`include "cu.v"
module gcd ();

reg [4:0] x,y;
wire [4:0] z,zr,xr,yr,tempr,Z;
reg clk,reset;
wire a,b,c;


initial begin
        $dumpfile("gcd_o.vcd");
    $dumpvars(0,gcd);
        clk=0;
        x=24;y=16; //input numbers
//#65 $finish;
end

always #5 clk=~clk;

dp first (z,x,y,xr,yr,tempr,reset,clk,c,zr,Z,a,b);
cu second (xr>0, xr>=yr,reset,clk, z,a,b,c);

endmodule
```

<u>Note:</u> Here, instead of integrating the wrapper unit for datapath and control unit, I've created a new module again to skip the testbench file as the inputs can be directly change in the wrapper code itself. Also multiple inputs can be given by suitable using the reset command.

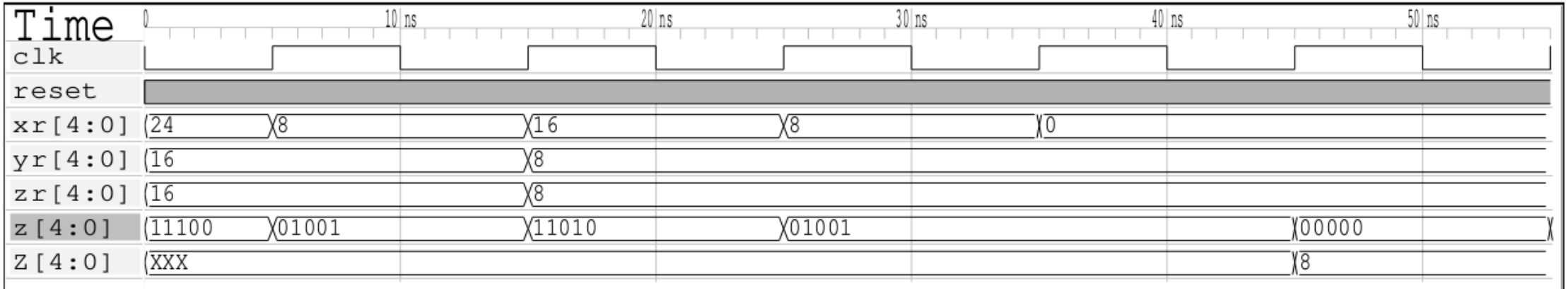
GCD (27,19) = 1



Here,

clk	=> clock
reset	=> reset signal
xr	=> x- register
Yr	=> y- register
Z	=> output signals from control unit
Z	=> Final output

GCD (24,16) = 8



Here,

clk	=> clock
reset	=> reset signal
xr	=> x- register
Yr	=> y- register
Z	=> output signals from control unit
Z	=> Final output