UNIVERSITY OF TEHRAN

Report Computer Assignment 1

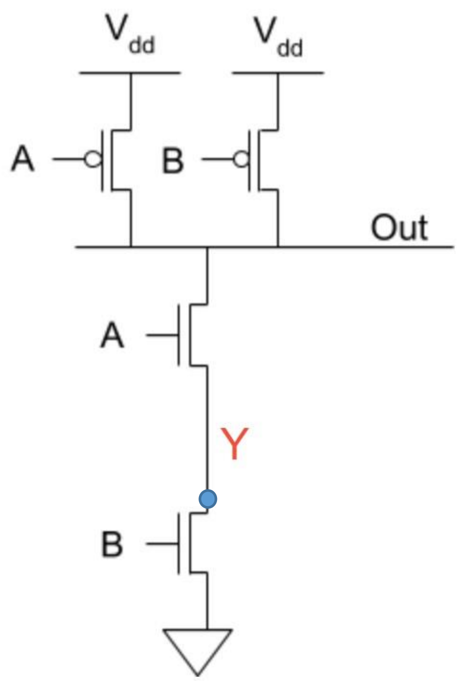
Digital Logic Design, ECE 367 / Digital Systems I, ECE 894

Danial Saeedi

Problem 1

Circuit Diagram

Here’s the circuit that I’m analyzing:



NAND CMOS Circuit

Verilog Code

Here’s my Verilog Code for Switch Level NAND Gate. (I used sublime for coloring this code).

The y wire is shown in the Circuit Diagram. a and b are the input of NAND Gate and the output is w :

```
`timescale 1ns/1ns
module MyNAND (input a,b,output w);
    wire y;
    supply1 Vdd;
    supply0 Gnd;
    pmos #(5,6,7) T1(w,Vdd,a);
    pmos #(5,6,7) T2(w,Vdd,b);
    nmos #(3,4,5) T3(y,Gnd,b);
    nmos #(3,4,5) T4(w,y,a);
endmodule
```

Worst-case Delay

To 1 : It takes 5ns to change the output of the PMOS transistors to 1. It takes 5 + 5 = 10ns to get Hi-Z from pull-down structure. Thus it takes **10ns** in the worst case to change the output of the NAND Gate to 1.

To 1 = 10ns

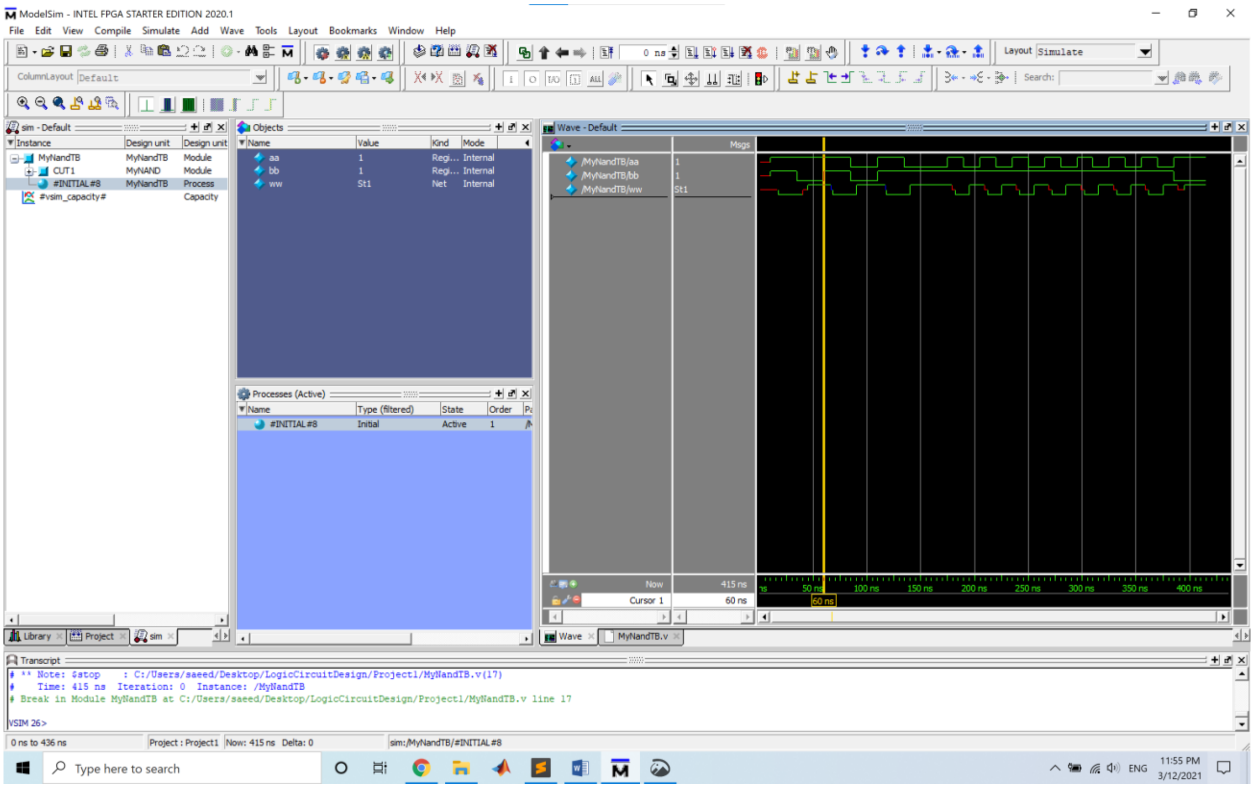
To 0 : It takes 7ns to change the output of pull-up structure to Hi-Z. It takes 4 + 4 = 8ns to change the output of pull-down structure to 0

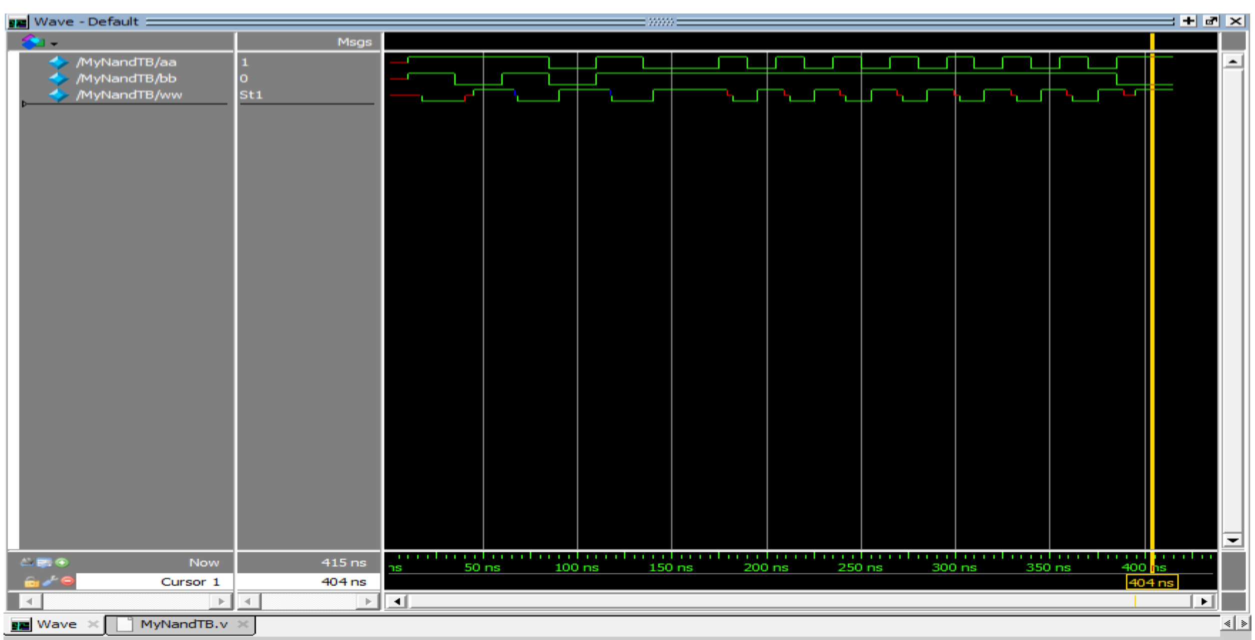
To 0 = 8ns

Testbench

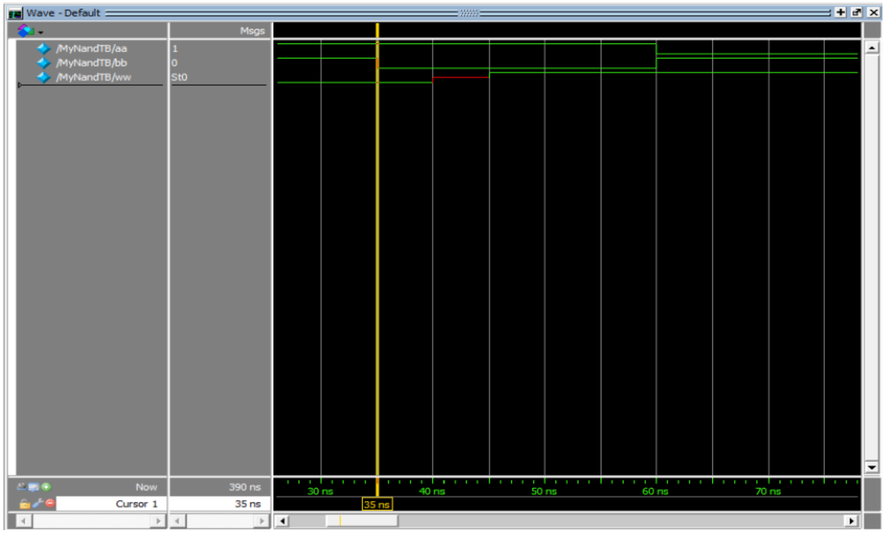
```
`timescale 1ns/1ns
module MyNandTB();
    //Inputs
    reg aa,bb;
    //The NAND output
    wire ww;
    MyNAND CUT1(aa,bb,ww);
    initial begin
    #10 aa = 1;bb = 1;
    #25 aa = 1;bb = 0;
    #25 aa = 1;bb = 1;
    #25 aa = 0;bb = 0;
    #25 aa = 1;bb = 1;
    #25 aa = 0;bb = 1;
    #25
    repeat(15) #15 aa = ~aa; bb = ~bb;
    #30 $stop;
    end
endmodule
```

Simulation Result

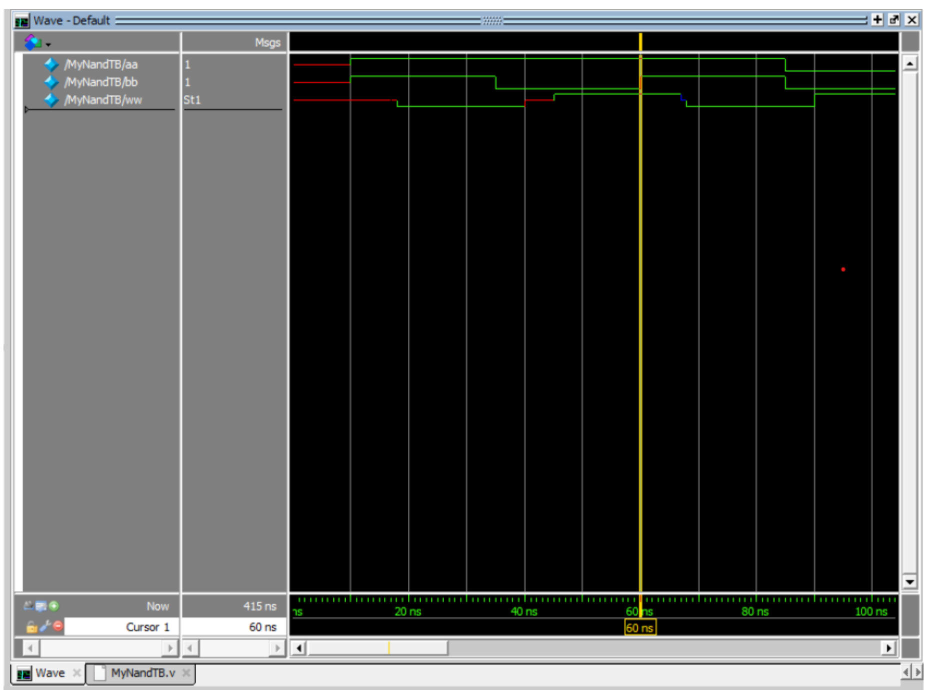




The To1 worst case happens when the input changes from a = 1 and b = 1 to a = 1 and b = 0.We expect to see 10ns delay. We can see from the zoomed waveform, To1 delay is correct: (The output changes to 1 at 45ns)



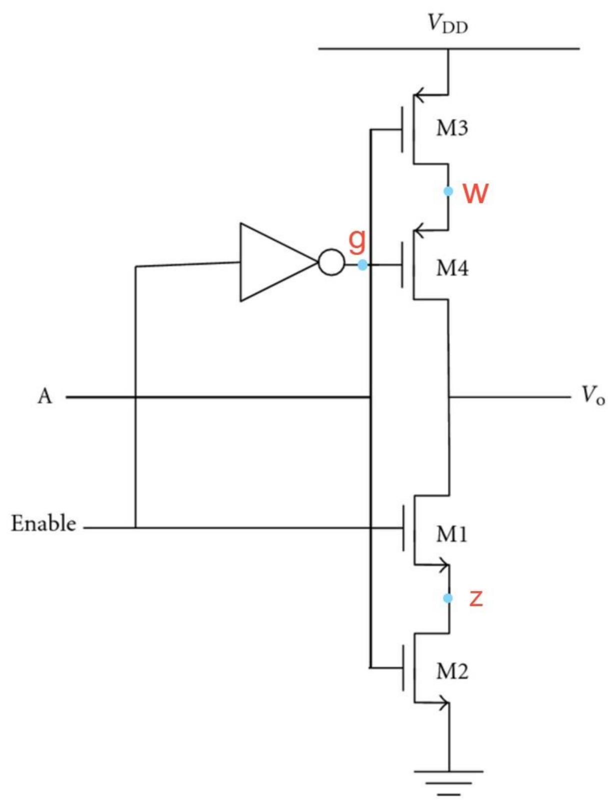
The To0 worst-case delay happens when the inputs changes from a = 1 and b = 0 to a = 1 and b = 1.As we expect, the To0 delay is 8ns:



Problem 2

Circuit Diagram

Here's the circuit that I'm analyzing:



Verilog Code

The y, w and z wires are shown in the Circuit Diagram. I use the inverter module that I created in part 1.

```
`timescale 1ns/1ns
module MyTriStateBuffer (input a,en,output y);
    wire g,w,z;
    supply1 Vdd;
    supply0 Gnd;
    MyInverter InverterGate(en,g);
    pmos #(5,6,7) T1(w,Vdd,a);
    pmos #(5,6,7) T2(y,w,g);
    nmos #(3,4,5) T3(y,z,en);
    nmos #(3,4,5) T4(z,Gnd,a);
endmodule
```

Testbench

```verilog
`timescale 1ns/1ns
module MyTriStateTB();
    wire yy;
    reg aa,enable;
    MyTriStateBuffer CUT1(aa,enable,yy);
    initial begin
    #10 aa = 1;enable = 0;
    #21 aa = 0;enable = 1;
    #21 aa = 1;enable = 1;
    #21 aa = 0;enable = 0;
    #21 aa = 1;enable = 1;
    #20
    repeat(15) #15 aa = ~aa; enable = ~enable;
    #30 $stop;
    end
endmodule
```

Worst-case Delay

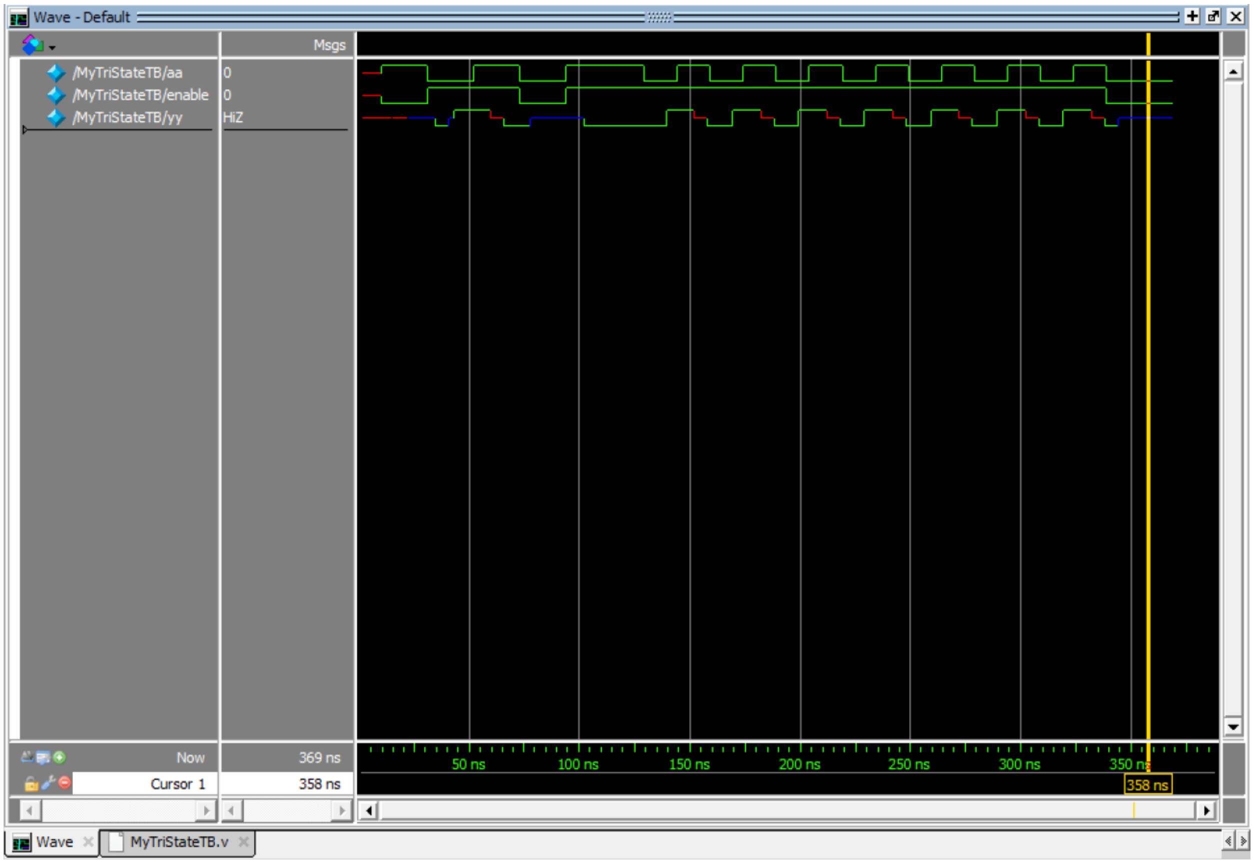
To 1 : It takes 6 + 5 = 11ns to changes the output of the pull-up structure to 1.(NOT gate delay is included)

To 1 = 11ns

To 0 : It takes 7 + 7 = 14ns to changes the output of the pull-up structure to Hi-Z. And it takes 4 + 4 = 8ns to to changes the output of the pull-down structure to 0.

To 0 = 14ns

Simulation Result

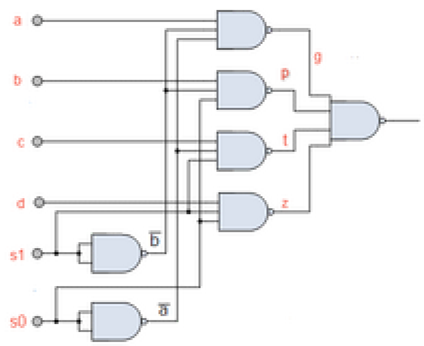


As expected, the To 1 and To 0 are 11ns and 14ns respectively.

Problem 3

Circuit Diagram

Here's the circuit that I'm analyzing:



Approach 1

From part 1, we know that NAND gate delays are #(10,8). We use NAND from VerilogSystem for this approach.

Verilog Code For Approach 1

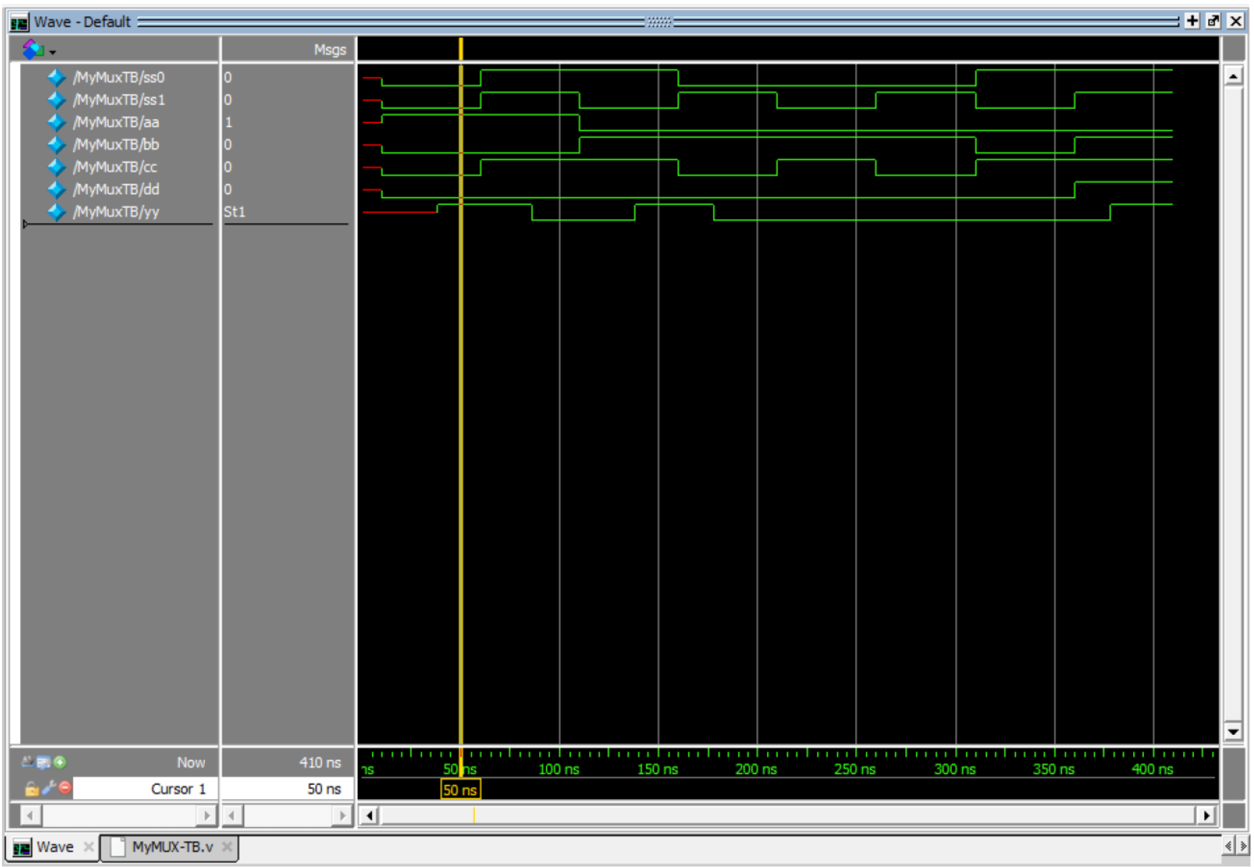
The s0_not,s1_not, g,p,t and z wires are shown in the circuit diagram.

```
`timescale 1ns/1ns
module MyMUX(
    input s0,s1,a,b,c,d,
    output y
);
    wire s0_not,s1_not,g,p,t,z;
    nand #(10,8) nand_1(s0_not,s0,s0);
    nand #(10,8) nand_2(s1_not,s1,s1);
    nand #(10,8) nand_3(g,a,s1_not,s0_not);
    nand #(10,8) nand_4(p,b,s1_not,s0);
    nand #(10,8) nand_5(t,c,s1,s0_not);
    nand #(10,8) nand_6(z,d,s1,s0);
    nand #(10,8) nand_7(y,g,p,t,z);
endmodule
```

Testbench For Approach 1

```
`timescale 1ns/1ns
module MyMuxTB();
    wire yy;
    reg ss0,ss1,aa,bb,cc,dd;
    MyMUX CUT1(ss0,ss1,aa,bb,cc,dd,yy);
    initial begin
    #10 ss1 = 0; ss0 = 0; aa = 1;bb = 0; cc = 0;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 1;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 0; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 1;
    #50 $stop;
    end
endmodule
```

Simulation Result For Approach 1



Analyzing Simulation Result

The output of MUX 4 to 1 is the result we expect.

For instance when the inputs are :

ss1 = 0; ss0 = 0; aa = 1;bb = 0; cc = 0;dd = 0;

We expect the output be 1. After 28ns delay,the output changes to 1.

Approach 2

We use the NAND gate module from part 1. I’ve created three NAND gates with 2, 3 and 4 inputs.

Note that this approach is much more accurate than approach 1. Because we’re simulating in transistor level.

Verilog Code for 3 and 4 inputs NAND gate :

Three Inputs NAND Gate

```
`timescale 1ns/1ns
module ThreeInputsNand(input a,b,c,output w);
    wire g,z;
    supply1 Vdd;
    supply0 Gnd;
    pmos #(5,6,7) T1(w,Vdd,a);
    pmos #(5,6,7) T2(w,Vdd,b);
    pmos #(5,6,7) T3(w,Vdd,c);
    nmos #(3,4,5) T4(w,z,a);
    nmos #(3,4,5) T5(z,g,b);
    nmos #(3,4,5) T6(g,Gnd,c);
endmodule
```

Four Inputs NAND Gate

```
`timescale 1ns/1ns
module ThreeInputsNand(input a,b,c,d,output w);
    wire g,p,z;
    supply1 Vdd;
    supply0 Gnd;
    pmos #(5,6,7) T1(w,Vdd,a);
    pmos #(5,6,7) T2(w,Vdd,b);
    pmos #(5,6,7) T3(w,Vdd,c);
    pmos #(5,6,7) T4(w,Vdd,d);
    nmos #(3,4,5) T5(w,z,a);
    nmos #(3,4,5) T6(z,g,b);
    nmos #(3,4,5) T7(g,p,c);
    nmos #(3,4,5) T8(p,Gnd,d);
endmodule
```

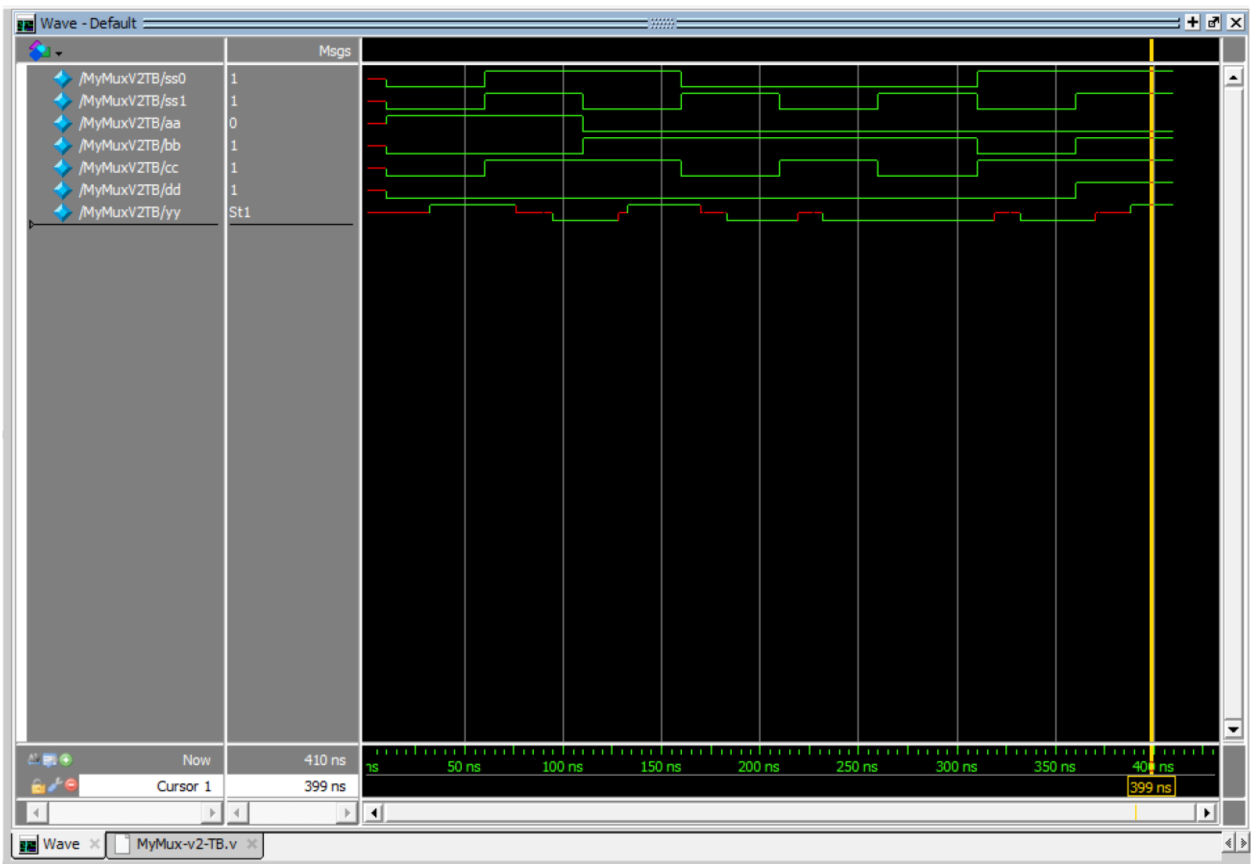
Verilog Code For Approach 2

```
`timescale 1ns/1ns
module MyMuxV2(
    input s0,s1,a,b,c,d,
    output y
);
    wire s0_not,s1_not,g,p,t,z;
    MyNAND nand_1(s0,s0,s0_not);
    MyNAND nand_2(s1,s1,s1_not);
    ThreeInputsNand nand_3(a,s1_not,s0_not,g);
    ThreeInputsNand nand_4(b,s1_not,s0,p);
    ThreeInputsNand nand_5(c,s1,s0_not,t);
    ThreeInputsNand nand_6(d,s1,s0,z);
    FourInputsNand nand_7(g,p,t,z,y);
endmodule
```

Testbench For Approach 2

```verilog
`timescale 1ns/1ns
module MyMuxV2TB();
    wire yy;
    reg ss0,ss1,aa,bb,cc,dd;
    MyMuxV2 CUT1(ss0,ss1,aa,bb,cc,dd,yy);
    initial begin
    #10 ss1 = 0; ss0 = 0; aa = 1;bb = 0; cc = 0;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 1;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 0; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 1;
    #50 $stop;
    end
endmodule
```

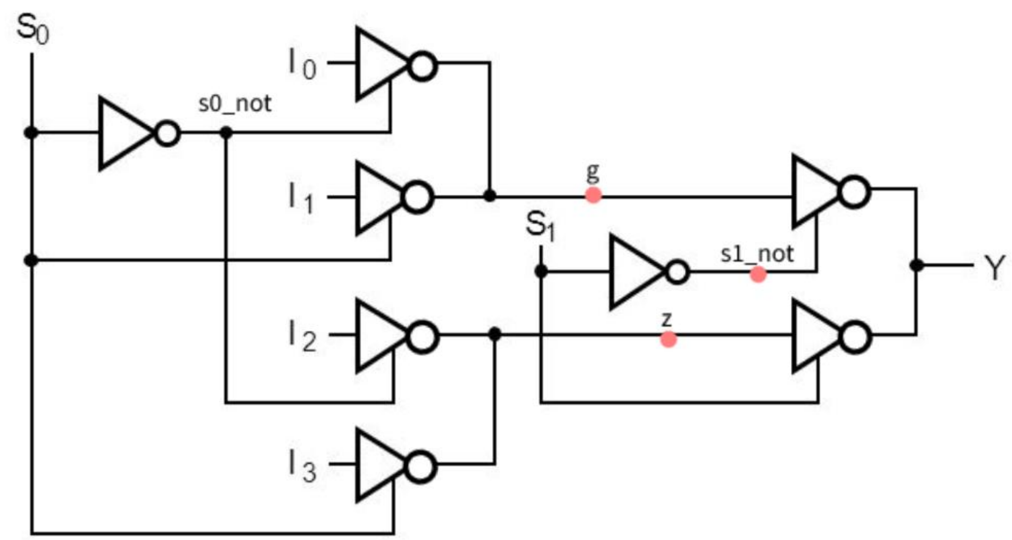
Simulation Result For Approach 2



Problem 4

Circuit Diagram

Here’s the circuit that I’m analyzing:



Verilog Code

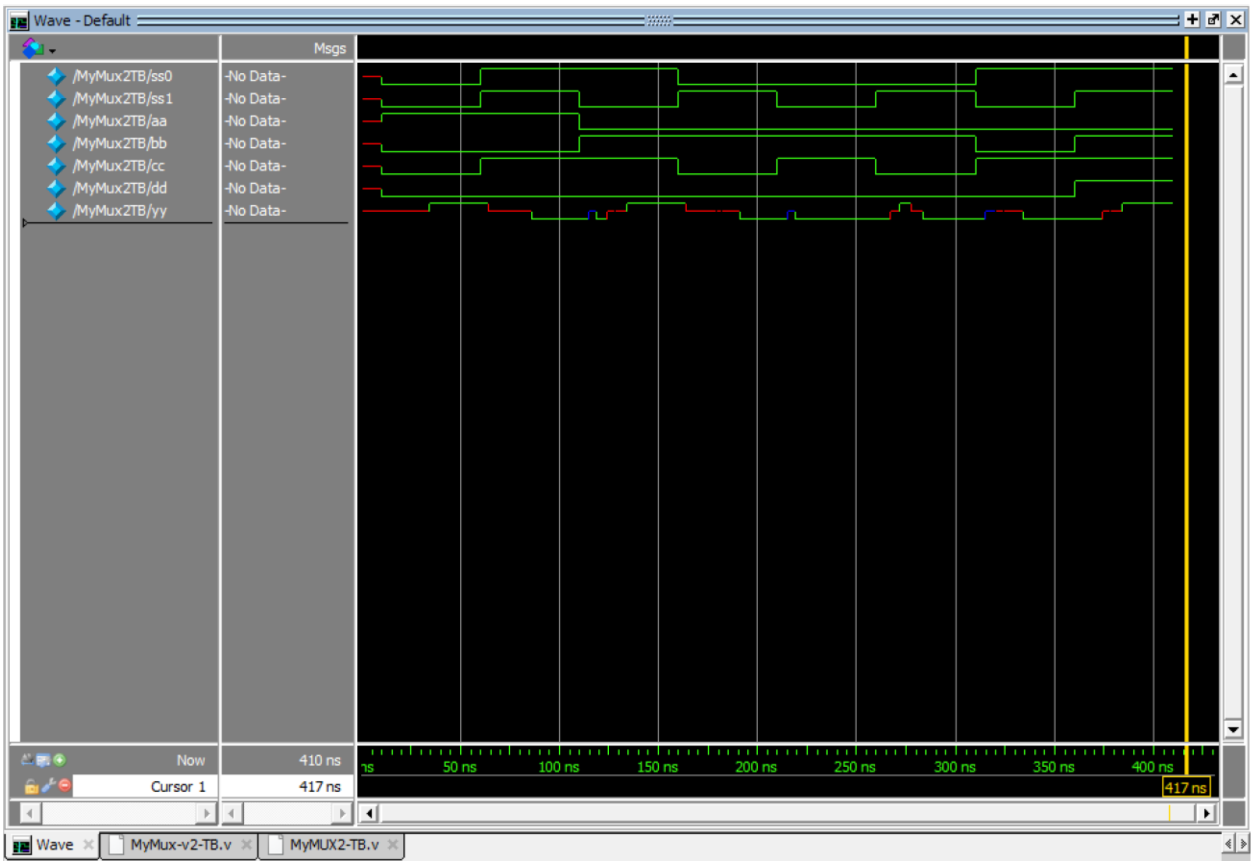
The s0_not, s1_not, g and z wires are shown in the circuit diagram.I used MyInverter module for the NOT gate and MyTriStateBuffer for the tri-state buffer gate.

```
`timescale 1ns/1ns
module MyMUX2(
    input s0,s1,a,b,c,d,
    output y
);
    wire s0_not,s1_not,g,z;
    MyInverter inverter_gate1(s0,s0_not);
    MyInverter inverter_gate2(s1,s1_not);

    MyTriStateBuffer not_buffer1(a,s0_not,g);
    MyTriStateBuffer not_buffer2(b,s0,g);
    MyTriStateBuffer not_buffer3(c,s0_not,z);
    MyTriStateBuffer not_buffer4(d,s0,z);

    MyTriStateBuffer not_buffer5(g,s1_not,y);
    MyTriStateBuffer not_buffer6(z,s1,y);
endmodule
```

Testbench

```
`timescale 1ns/1ns
module MyMux2TB();
    wire yy;
    reg ss0,ss1,aa,bb,cc,dd;
    MyMUX2 CUT1(ss0,ss1,aa,bb,cc,dd,yy);
    initial begin
    #10 ss1 = 0; ss0 = 0; aa = 1;bb = 0; cc = 0;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 1;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 0; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 1;
    #50 $stop;
    end
endmodule
```

Simulation Result



Problem 5

Testbench

```verilog
`timescale 1ns/1ns
module MUXComparison();
    wire mux1_output,mux2_output;
    reg ss0,ss1,aa,bb,cc,dd;
    //Mux Problem 3
    MyMuxV2 mux1(ss0,ss1,aa,bb,cc,dd,mux1_output);
    //Mux Problem 4
    MyMUX2 mux2(ss0,ss1,aa,bb,cc,dd,mux2_output);
    initial begin
    #10 ss1 = 0; ss0 = 0; aa = 1;bb = 0; cc = 0;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 1;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 0; aa = 0;bb = 1; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 0; aa = 0;bb = 1; cc = 0;dd = 0;
    #50 ss1 = 0; ss0 = 1; aa = 0;bb = 0; cc = 1;dd = 0;
    #50 ss1 = 1; ss0 = 1; aa = 0;bb = 1; cc = 1;dd = 1;
    #50 $stop;
    end
endmodule
```

Simulation Result

Calculating the number of transistors from problem 3

Every two-input NAND gate needs 2 PMOS and 2 NMOS transistors.(Total : 4)

Every three-input NAND gate needs 3 PMOS and 3 NMOS transistors.(Total : 6)

Every four-input NAND gate needs 4 PMOS and 4 NMOS transistors.(Total : 8)

There are 2 two-inputs NAND gate and 4 three-inputs NAND gate and 1 four-inputs NAND gate. Thus the total transistors we need for this problem is :

$$2*4+4*6+1*8 = 40 \text{ transistors}$$

Calculating the number of transistors from problem 4

Every NOT gate needs 1 PMOS and 1 NMOS transistors.(Total : 2)

Every tri-state buffer gate needs 6 transistors.

There are 2 NOT gate and 6 tri-state buffer gate. Thus the total transistors we need for this problem is :

$$6*6+2*2 = 40 \text{ transistors}$$

Cost And Power Consumption

Both problems need 40 transistors. So the cost is the same.But problem 3 and 4 need 7 and 8 gates respectively. Each gate is connected to supply 1 and supply 0 in its CMOS structure.

So problem 4 consumes more power than problem 3.