

CA 2

Amirali Mokhtarian

810803088

1)

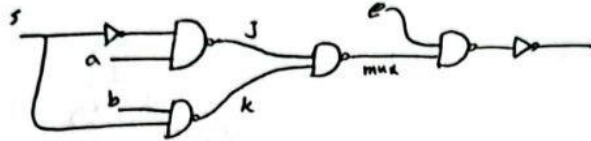
```

MUX.v
1  `timescale 1ns/1ns
2
3  module Nand2(input a,b , output w);
4      wire j;
5      supply1 vdd; supply0 gnd;
6      pmos #(5,6,7) p1(w,vdd,a);
7      pmos #(5,6,7) p2(w,vdd,b);
8
9      nmos #(3,4,5) n1(j,gnd,b);
10     nmos #(3,4,5) n2(w,j,a);
11 endmodule
12
13 module inverter(input a, output abar);
14     supply1 vdd; supply0 gnd;
15     pmos #(5,6,7) T1(abar,vdd,a);
16     nmos #(3,4,5) T2(abar,gnd,a);
17 endmodule
18
19 module mux_e(input a,b,s,e, output y);
20     wire sbar;
21     wire j, k;
22     wire mux;
23     wire ybar;
24
25     inverter no(s,sbar);
26
27     Nand2 N1( a, sbar, j);
28     Nand2 N2(s, b, k);
29     Nand2 N3(k,j,mux);
30     Nand2 N4(e, mux, ybar);
31
32     inverter no2(ybar,y);
33 endmodule

```

810803088

①



$$y = e ? (a \& \sim s) | (b \& s) : 1'b0$$

Worst case delay to 1:

a	b	s	e	→	a b s e
1	0	1	1	→	1 0 0 1

not # (5, 7)

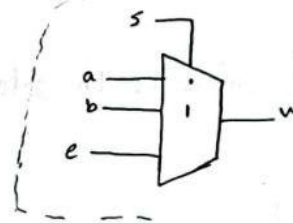
nand # (10, 8)

$$5 + 8 + 10 + 8 + 5 = 36 ns$$

Worst case delay to 0:

1 0 0 1 → 1 0 1 1

$$7 + 10 + 8 + 10 + 7 = 42 ns$$



2)

assignMUX.v

1 `timescale 1ns/1ns

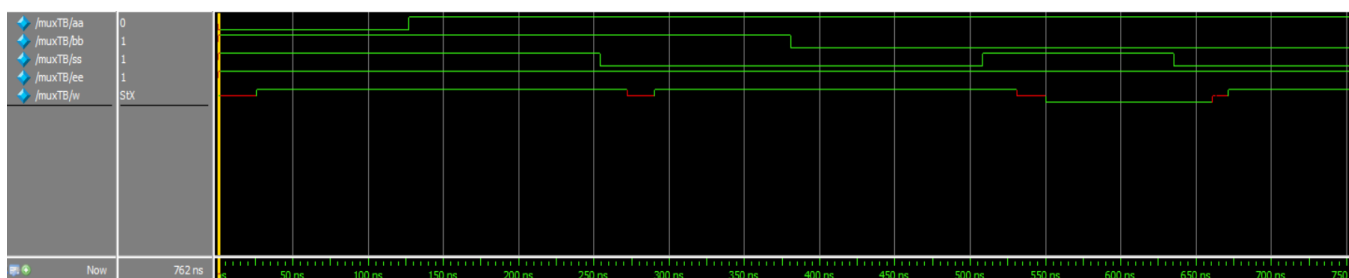
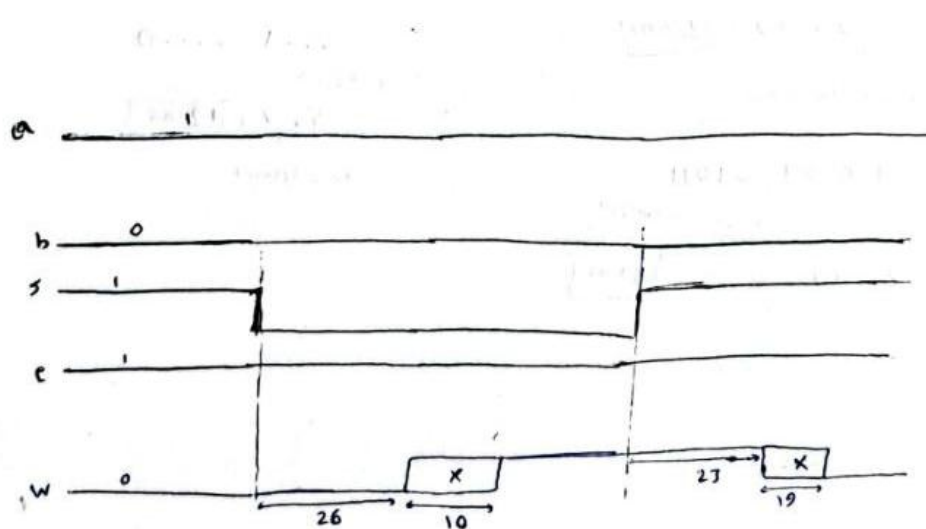
2 module assign_MUX(input a,b,s,e , output w);

3 | assign #(36,42) w = e?(a&~s)|(b&s):1'b0;

4 endmodule

Test bench 1,2)

```
≡ muxTB.v
1  `timescale 1ns/1ns
2
3  module muxTB();
4      reg aa=1'b0 , bb=1'b1 , ss=1'b1 , ee=1'b1;
5      wire w,assign_w;
6      mux_e mt(aa,bb,ss,ee,w);
7      assign_MUX asmt(aa,bb,ss,ee,assign_w);
8      initial begin
9          □ #127;
10         {aa,bb,ss,ee}=4'b1111; □ #127;//to0
11         {aa,bb,ss,ee}=4'b1101; □ #127;//X interval
12
13         {aa,bb,ss,ee}=4'b1001; □ #127;
14
15         {aa,bb,ss,ee}=4'b1011; □ #127;//worst to0
16         {aa,bb,ss,ee}=4'b1001; □ #127;//worst to1
17
18         $finish;
19     end
20 endmodule
```



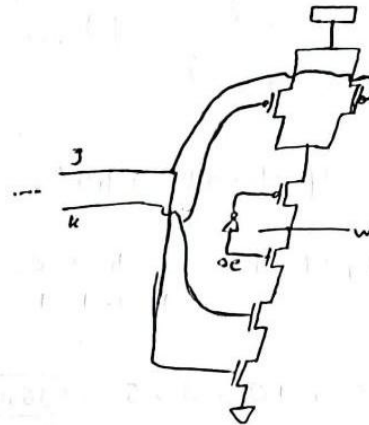
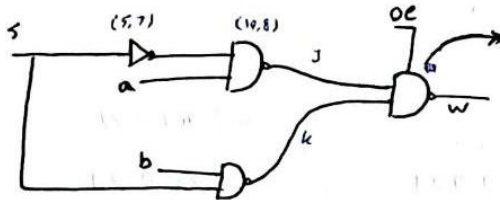
The difference between the path delays is larger than the delay of the last gate (the final NAND), and since the code is at the transistor level, there is no glitch (hazard) in the waveform. In the transistor-level circuit, we have a *toZ* delay, so it's not possible for a gate to output 1 (or 0) when it should be propagating 0 (or 1). In this case, we see a Z state instead of a glitch. (X in the wave)

3)

```
muxBUF.v
1  `timescale 1ns/1ns
2
3  module Nand2(input a,b , output w);
4      wire j;
5      supply1 vdd; supply0 gnd;
6      pmos #(5,6,7) p1(w,vdd,a);
7      pmos #(5,6,7) p2(w,vdd,b);
8
9      nmos #(3,4,5) n1(j,gnd,b);
10     nmos #(3,4,5) n2(w,j,a);
11 endmodule
12
13
14 module inverter(input a, output abar);
15     supply1 vdd; supply0 gnd;
16     pmos #(5,6,7) T1(abar,vdd,a);
17     nmos #(3,4,5) T2(abar,gnd,a);
18 endmodule
19
20 module muxBYbuf(input a,b,s,oe , output w);
21     wire sbar,j,k;
22     wire y,oebar;//for 3state nand
23
24     inverter i1(s,sbar);
25     Nand2 n1(a,sbar,j);
26     Nand2 n2(b,s,k);
27
28     Nand2 nMUX(k,j,y);
29     inverter i2(oe,oebar);
30     pmos #(5,6,7) p1_end(w,y,oebar);
31     nmos #(3,4,5) n1_end(w,y,oe);
32 endmodule
```

③ MUX with tri-state nand

$$y = oe ? (a \& \sim s) | (b \& s) : 1'bZ$$



* \bar{s} and \bar{J} must be the gates of the nearest transistor to the gnd. (to have the worst case)

worst case to 1: $a \quad b \quad s \quad e$
 $1 \quad 0 \quad 1 \quad 1 \rightarrow 1001$

$$5 + \overset{\text{to 0, find}}{8} + \underset{\text{to 1 delay of the buffer}}{5} + 10 = \boxed{28ns}$$

worst case to 0: $1001 \rightarrow 1011$

$$7 + 10 + \overset{\text{to 0 (for pmos of buffer)}}{8} + \underset{\text{to 0 (second nand)}}{6} = \boxed{31ns}$$

worst case
to Z:

$$\dots 1 \rightarrow \dots 0$$

$$\overset{\text{to 1 (inv)}}{5} + 7 = \boxed{12ns}$$

to Z (pmos)

4)

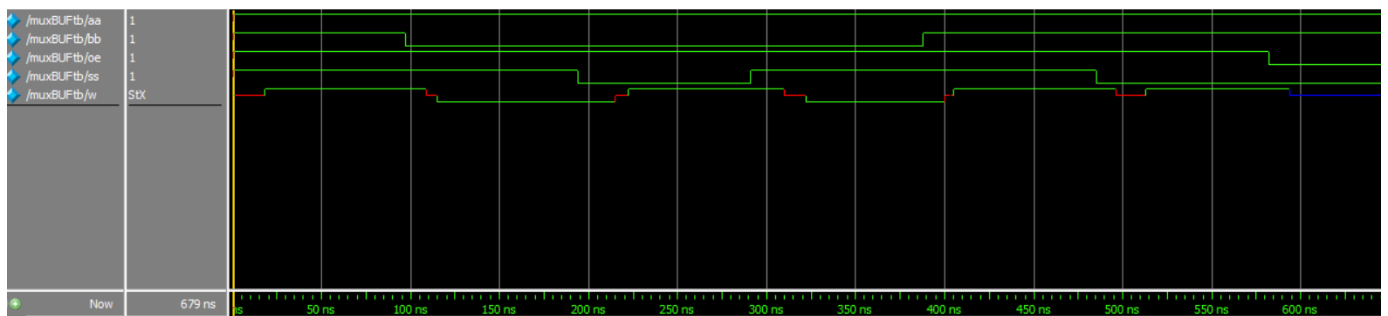
```

assignMUXbuf.v
1  `timescale 1ns/1ns
2  module assignMUXbuf(input a,b,s,oe , output w);
3      assign #(28,31,12) w = oe ? (a&~s) | (b&s) : 1'bZ;
4  endmodule

```

Test bench 3,4)

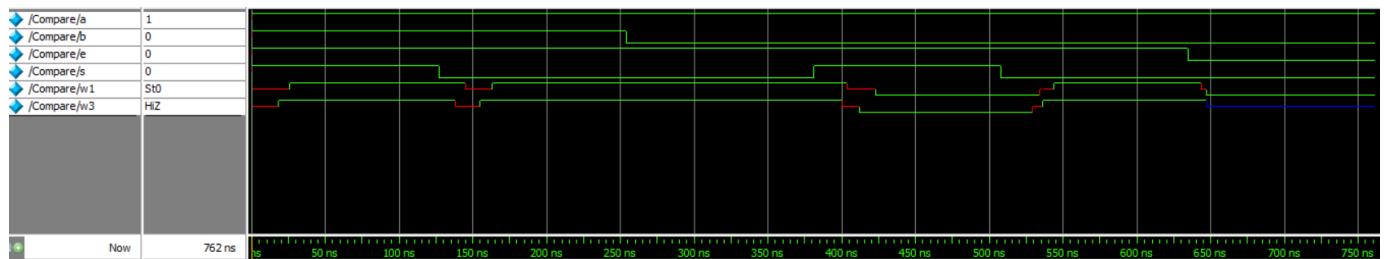
```
muxBUFTb.v
1  `timescale 1ns/1ns
2
3  module muxBUFTb();
4      reg aa=1'b1 , bb=1'b1 , ss=1'b1 , oe=1'b1;
5      wire w;
6      muxBYbuf mbt(aa,bb,ss,oe,w);
7      assignMUXbuf ambt(aa,bb,ss,oe,w_assign);
8      initial begin
9          #97;
10         {aa,bb,ss,oe}=4'b1011;#97;
11         {aa,bb,ss,oe}=4'b1001;#97;//worst case to1
12         {aa,bb,ss,oe}=4'b1011;#97;//worst case to0
13         {aa,bb,ss,oe}=4'b1111;#97;
14         {aa,bb,ss,oe}=4'b1101;#97;//X interval
15         {aa,bb,ss,oe}=4'b1100;#97;//toZ
16         $finish;
17     end
18 endmodule
```



Just like the testbench of 1&2, There's no glitch because of the transistor_level code and the difference between the path delays is larger than the delay of the last gate.

5)

```
CompareTB.v
1  `timescale 1ns/1ns
2  module Compare();
3      reg a=1'b1 , b=1'b1 , s=1'b1;
4      reg e=1'b1;
5      wire w1 , w3;
6
7      mux_e mux1(a,b,s,e,w1);
8      muxBYbuf mux_3(a,b,s,e,w3);
9
10     initial begin
11         #127;
12         {a,b,s,e}=4'b1101; #127; //X interval
13         {a,b,s,e}=4'b1001; #127;
14         {a,b,s,e}=4'b1011; #127; //worst to0
15         {a,b,s,e}=4'b1001; #127; //worst to1
16         {a,b,s,e}=4'b1000; #127; //Z for 3
17         $finish;
18     end
19 endmodule
```



The structure of 3rd problem circuit has less transistors and gates than the first one so a multiplexer with tri_state NAND has less delay than a multiplexer with normal NANDs.

Transistors in first problem (which named *mux_e*):

$$2+4+4+4+4+2 = \mathbf{20}$$

Transistors in third problem (which named *muxBYbuf*):

$$2+4+4+6+2 = \mathbf{18}$$

Power consumption of the first problem is more than the 3rd one because as I showed it has more transistors.