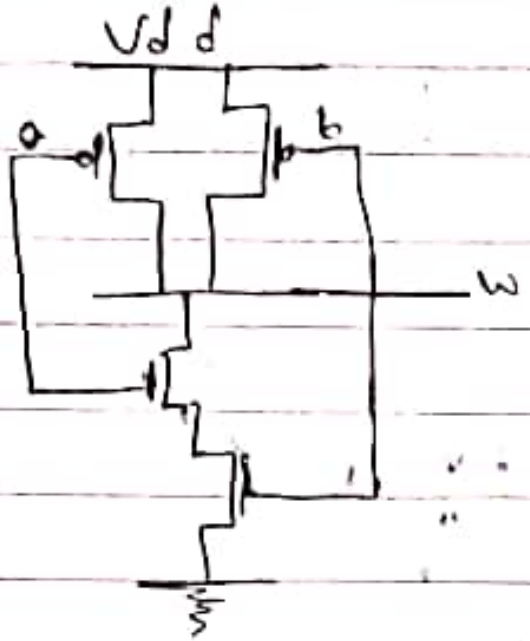
Digital systems _ Mahdi Bokhaal _ CA1 _ Report

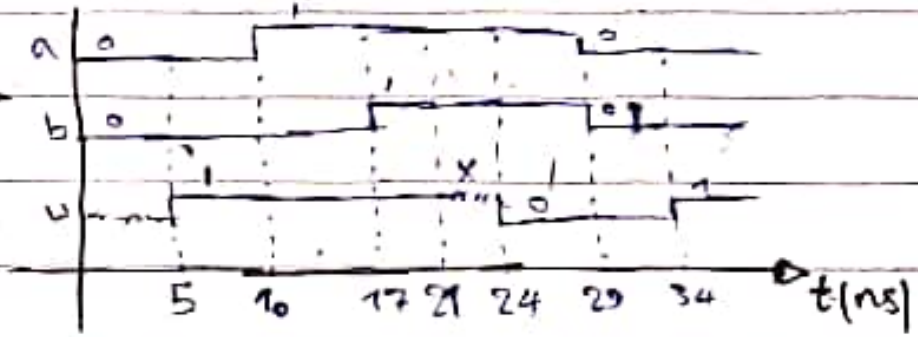
①



nMos # (3,4,5)

Pmos # (5,6,7)

Hand_Simulate → Init → a=0, b=0 →

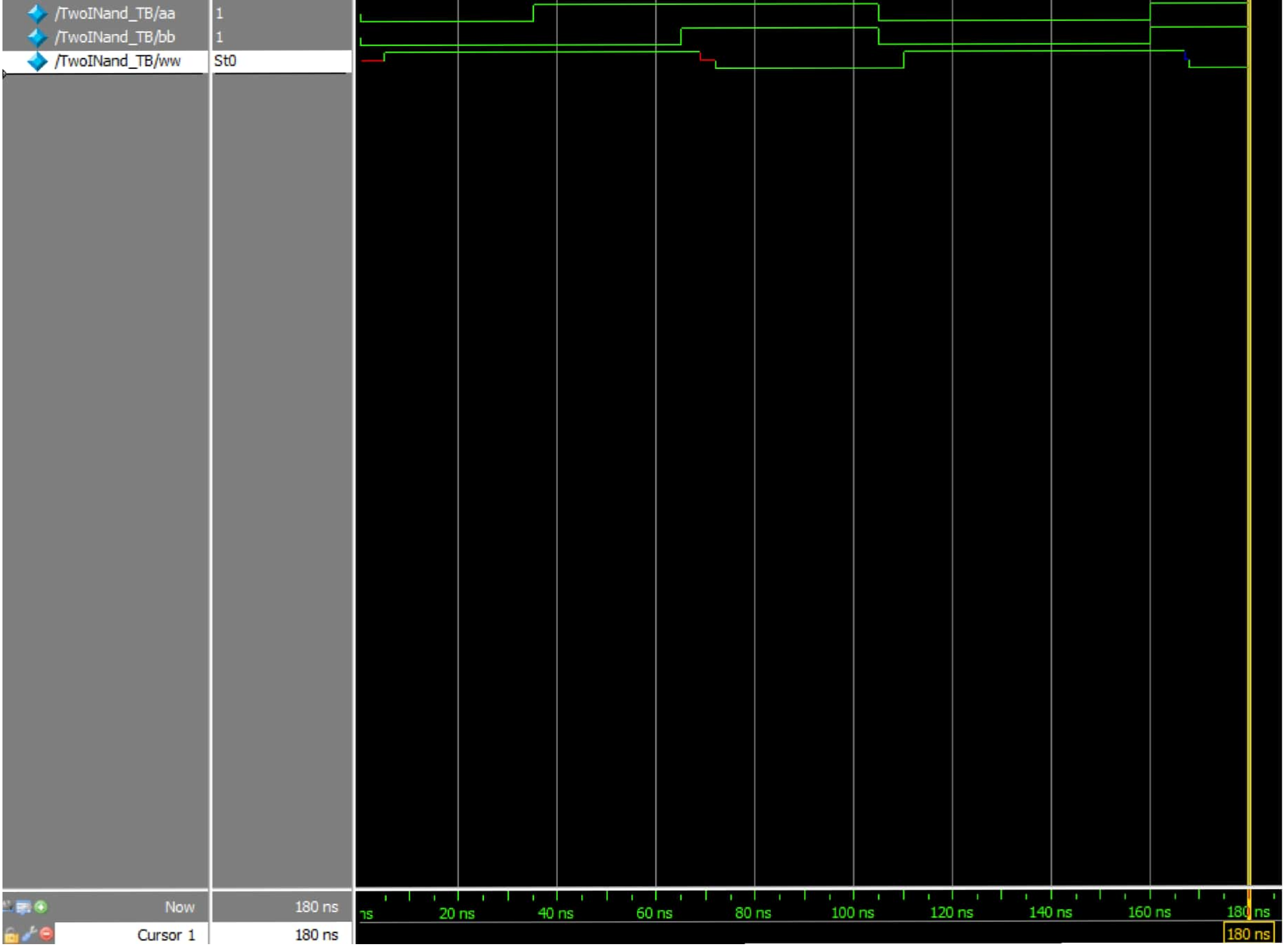


Two input nand

```
`timescale 1ns/1ns

module TwoINand(input a, b, output w);
    supply1 Vdd;
    supply0 Gnd;
    wire y1;
    pmos #(5,6,7) T1(w, Vdd, a), T2(w, Vdd, b);
    nmos #(3,4,5) T3(y1,Gnd,a), T4(w,y1,b);
endmodule
```

Two input nand testbench

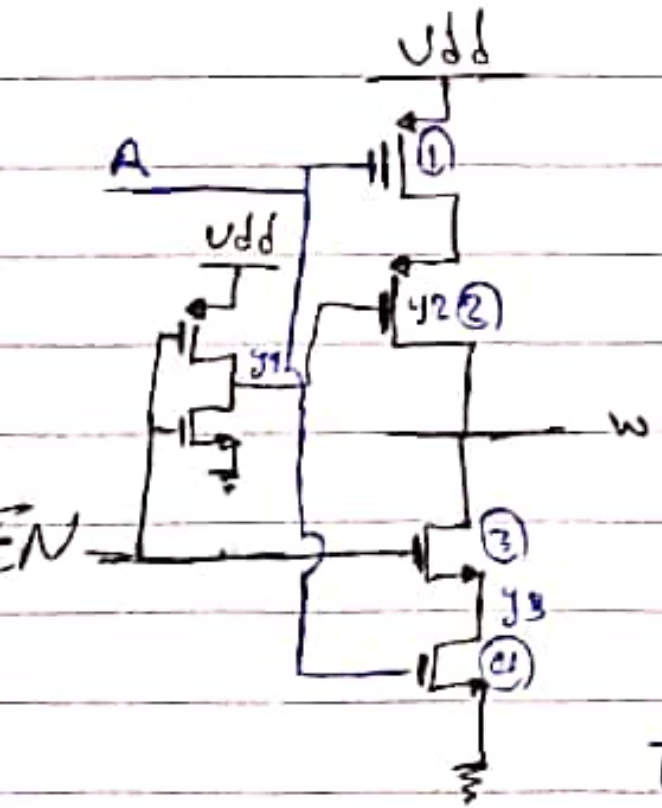
```
`include "TwoInputNand.v"

`timescale 1ns/1ns


module TwoINand_TB();
    reg aa=0,bb=0;
    wire ww;
    TwoINand two_nand(.a(aa), .b(bb), .w(ww));
    initial begin
        #15
        #20 aa=1;
        #30 bb=1;
        #40 aa=0;bb=0;
        #55 aa=1;bb=1;
        #20 $stop;
    end
endmodule
```

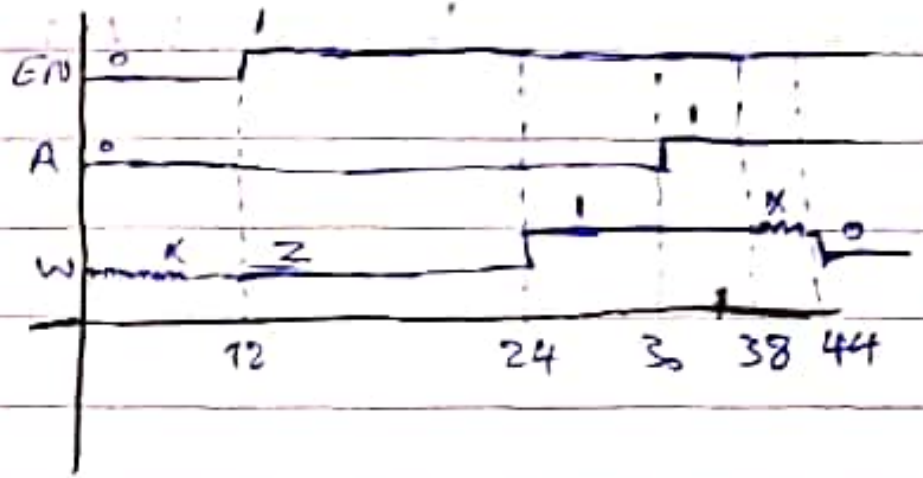


② : Tri_state buffer Schematic

EN	A	W
0	0	Z
0	1	Z
1	0	1
1	1	0

→ $W = EN.\bar{A}$





The worst case occurs when EN=1 and A turns from 0 to 1, transistor① takes 7ns for 1→Z, and transistor② has output 1 so far, takes 7ns.

And two nmos transistors take 8ns to give 0 as output. As a result, we will have

X in [14-8=6ns] → [worst case = 14ns] → To 0

To 1 → Assuming A=1, Ex turns to 1 from 0 → inverter delay: 7ns + Pass to 1 delay: 5

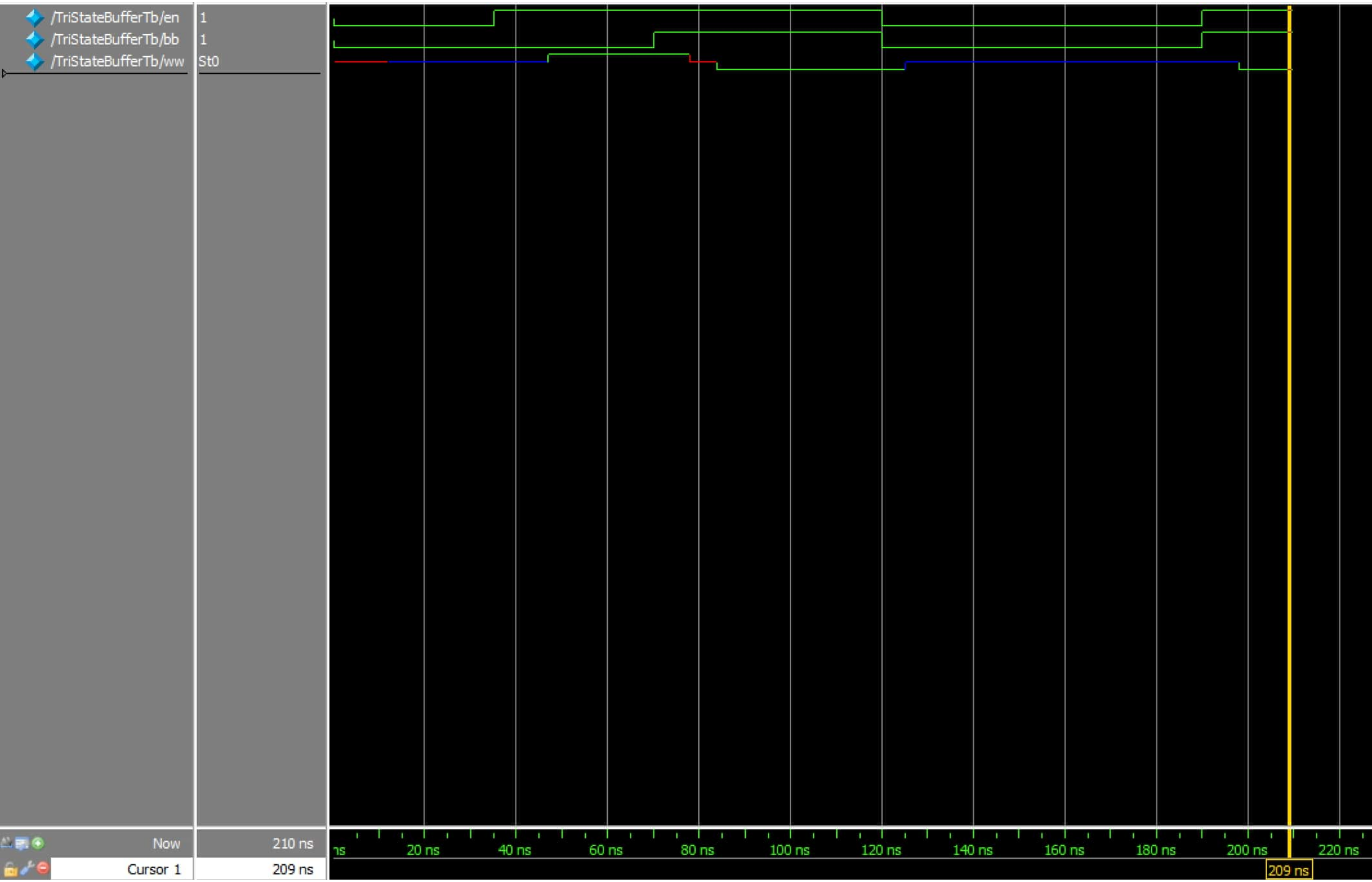
→ 7+5=12ns

Tri-State Buffer

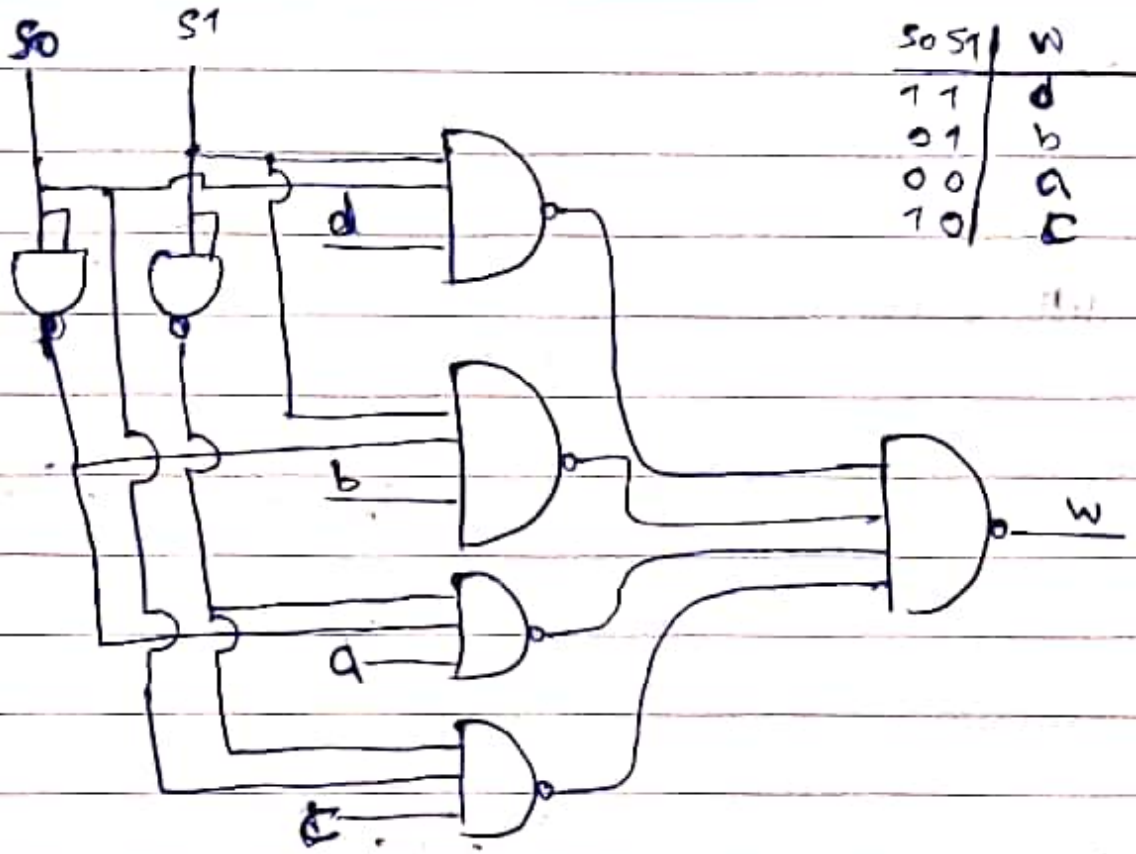
```
`timescale 1ns/1ns

module TriStateBuffer(input en, a, output w);
    supply1 Vdd;
    supply0 Gnd;
    wire y1, y2, y3;
    pmos #(5,6,7) T1(y1,Vdd,en), T3(y2,Vdd,a), T4(w,y2,y1);
    nmos #(3,4,5) T2(y1,Gnd,en), T5(y3,Gnd,a), T6(w,y3,en);
endmodule
```

Tri-State Buffer testbench

```
`include "TriStateBuffer.v"
`timescale 1ns/1ns


module TriStateBufferTb();
    reg en=0,bb=0;
    wire ww;
    TriStateBuffer tri_state(.en(en), .a(bb), .w(ww));
    initial begin
        #15
        #20 en=1;
        #35 bb=1;
        #50 en=0;bb=0;
        #70 en=1;bb=1;
        #20 $stop;
    end
endmodule
```



(3)



S0	S1	W
1	1	d
0	1	b
0	0	a
1	0	c

delay time for 2 input Nand gate → T01: 10ns , T00 : 8ns

" " " 3input Nand gate → T01, 15ns, T00 : 12ns

" " " 4input Nand gate → T01 : 20ns, T00 . 16ns

Three input nand

```
`timescale 1ns/1ns

module ThreeInputNand(input a,b,c, output w);
    supply1 Vdd;
    supply1 Gnd;

    wire y1,y2;
    pmos #(5,6,7) T1(w,Vdd,a), T2(w,Vdd,b), T3(w,Vdd,c);
    nmos #(3,4,5) T4(w,y2,a), T5(y2,y1,b), T6(y1,Gnd,c);
endmodule
```

Four input nand

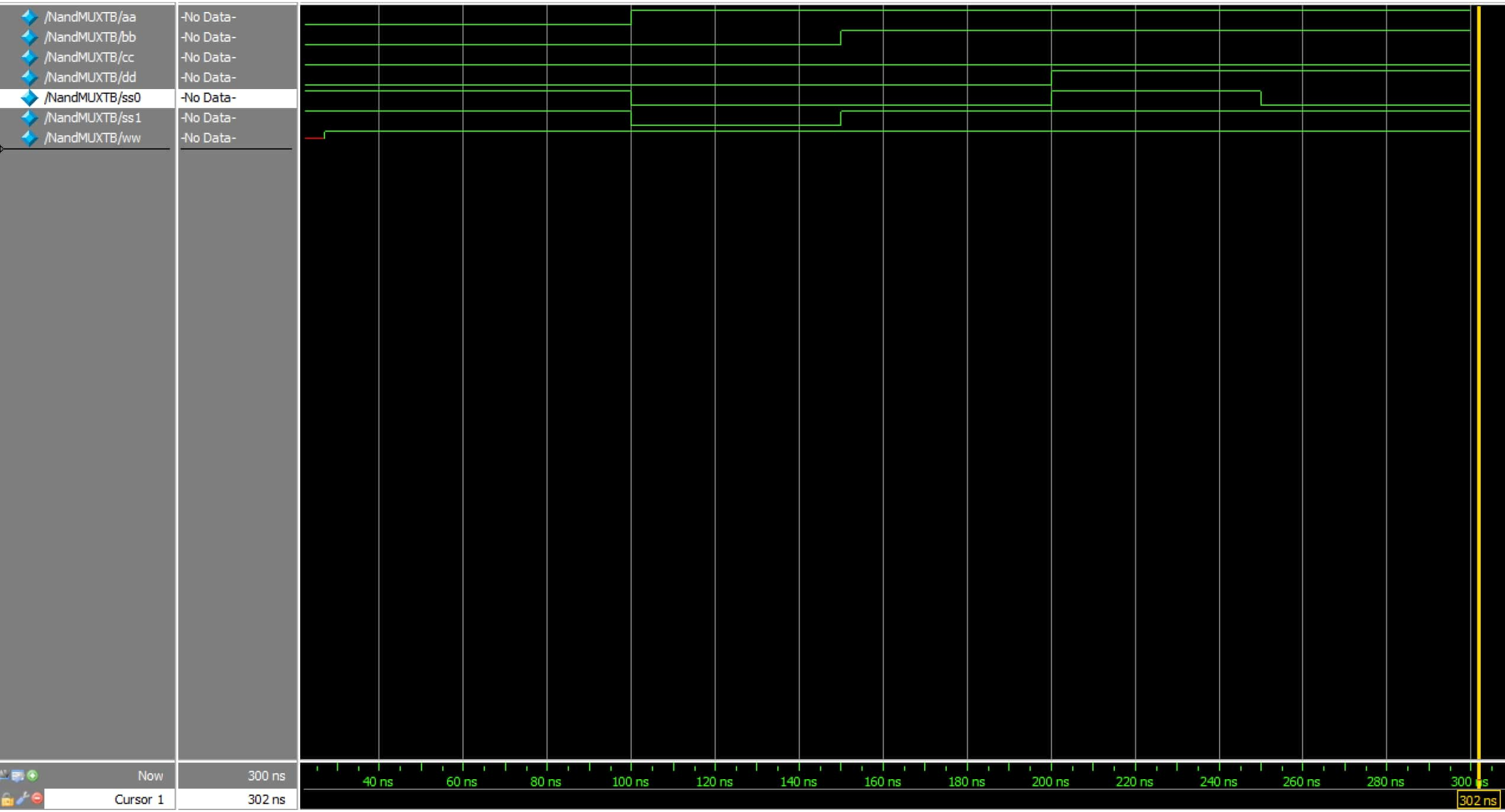
```
`timescale 1ns/1ns

module FourInputNand(input a,b,c,d, output w);
    supply1 Vdd;
    supply1 Gnd;

    wire y1,y2,y3;
    pmos #(5,6,7) T1(w,Vdd,a), T2(w,Vdd,b), T3(w,Vdd,c), T4(w,Vdd,d);
    nmos #(3,4,5) T5(w,y3,a), T6(y3,y2,b), T7(y2,y1,c), T8(y1,Gnd,d);
endmodule
```

Nand 4-To-1 MUX

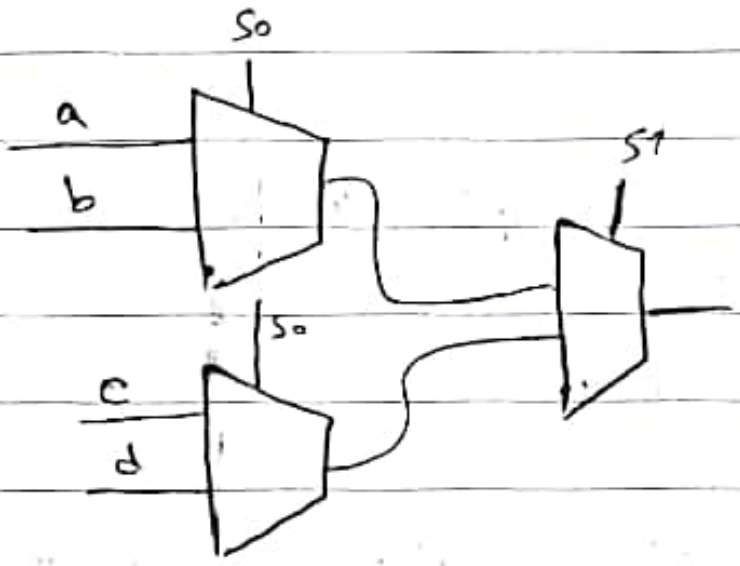
```
`include "TwoInputNand.v"
`include "ThreeInputNand.v"
`include "FourInputNand.v"


`timescale 1ns/1ns


module NandMUX(input a,b,c,d,s0,s1, output w);
    wire s0_bar, s1_bar;
    wire a_s, b_s, c_s, d_s;
    TwoINand invert_s0(s0,s0,s0_bar);
    TwoINand invert_s1(s1,s1,s1_bar);
    ThreeInputNand select_a(s0_bar, s1_bar, a, a_s);
    ThreeInputNand select_b(s0_bar, s1, b, b_s);
    ThreeInputNand select_c(s0, s1_bar, c, c_s);
    ThreeInputNand select_d(s0, s1, d, d_s);
    FourInputNand result(a_s, b_s, c_s, d_s, w);
endmodule
```

Nand 4-To-1 MUX testbench

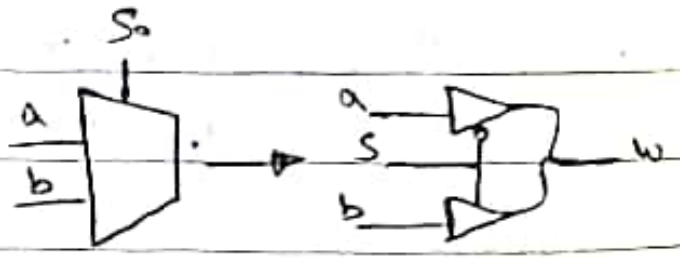
```
`include "4To1MUX.v"

`timescale 1ns/1ns

module NandMUXTB( );
    reg aa=0,bb=0,cc=0,dd=0,ss0=1,ss1=1;
    wire ww;
    NandMUX mux(aa,bb,cc,dd,ss0,ss1,ww);

    initial begin
        #50
        #50 aa=1; ss0=0; ss1=0;
        #50 bb=1; ss1=1;
        #50 dd=1; ss0=1; ss1=1;
        #50 ss0=0; ss1=1;
        #50 $stop;
    end
endmodule
```



(4): Convert 4_To_1 MUX to 3 2_To_1 MUX →



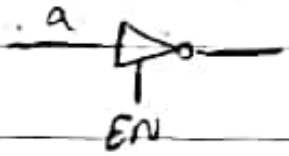
Now Create a 2_To_1 MUX with Tri_state buffer →



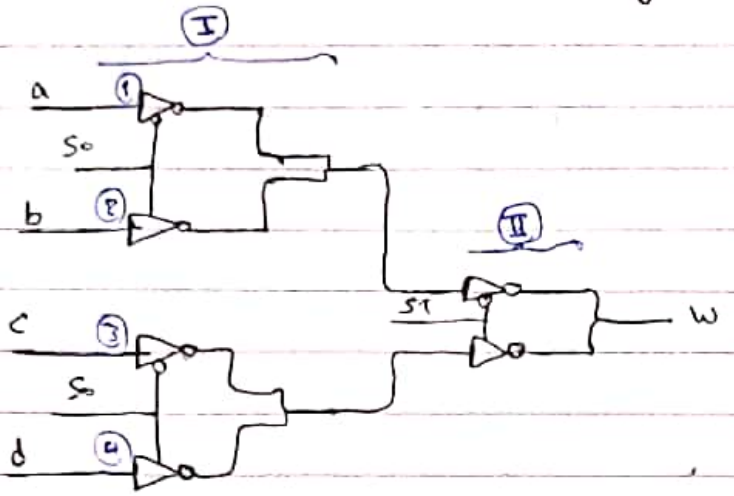
Truth table →

S0	w
0	a
1	b

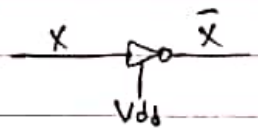
, Now according to question two we now the structure of our Tri_state buffer is as follow:

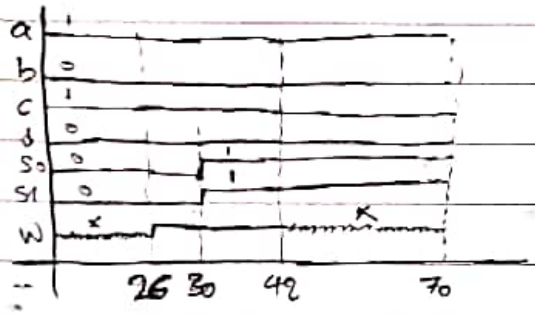


Now we create the 4_To_1 MUX using this structure →



Now, according to the drawn schematic, we know that the inverters of part (I) neutralize the inverters of Part (II) and there is no change in the output of our circuit.

If we want $\overline{S_0}$ or $\overline{S_1}$ → 



assume a=1 b=0 c=1 d=0

S0, S1 = 0

So: ∅ → 1 → delay time ① ③ 14 (Tri_state Inverter To∅) + 12 (Tri_state ① To2)
③
② ④ 12 (Tri-state To1)

→ 16ns → X

S1: ∅ → 1 → In the first 26 ns it looks like So and after that it becomes 1 as input →
14ns (Tri_state To∅)

→ worst_case delay → 40ns : To∅

To1 → So: 1 → ∅ delay time ① ⑤ → 12 (Tri_state Inverter To1) + 14 (Tri_state To∅)
② ④ 14 (Tri_state To∅)
→ 12ns → X

S1: 1 → ∅ → delay time → 26ns same as So and after that it becomes ∅ as input →

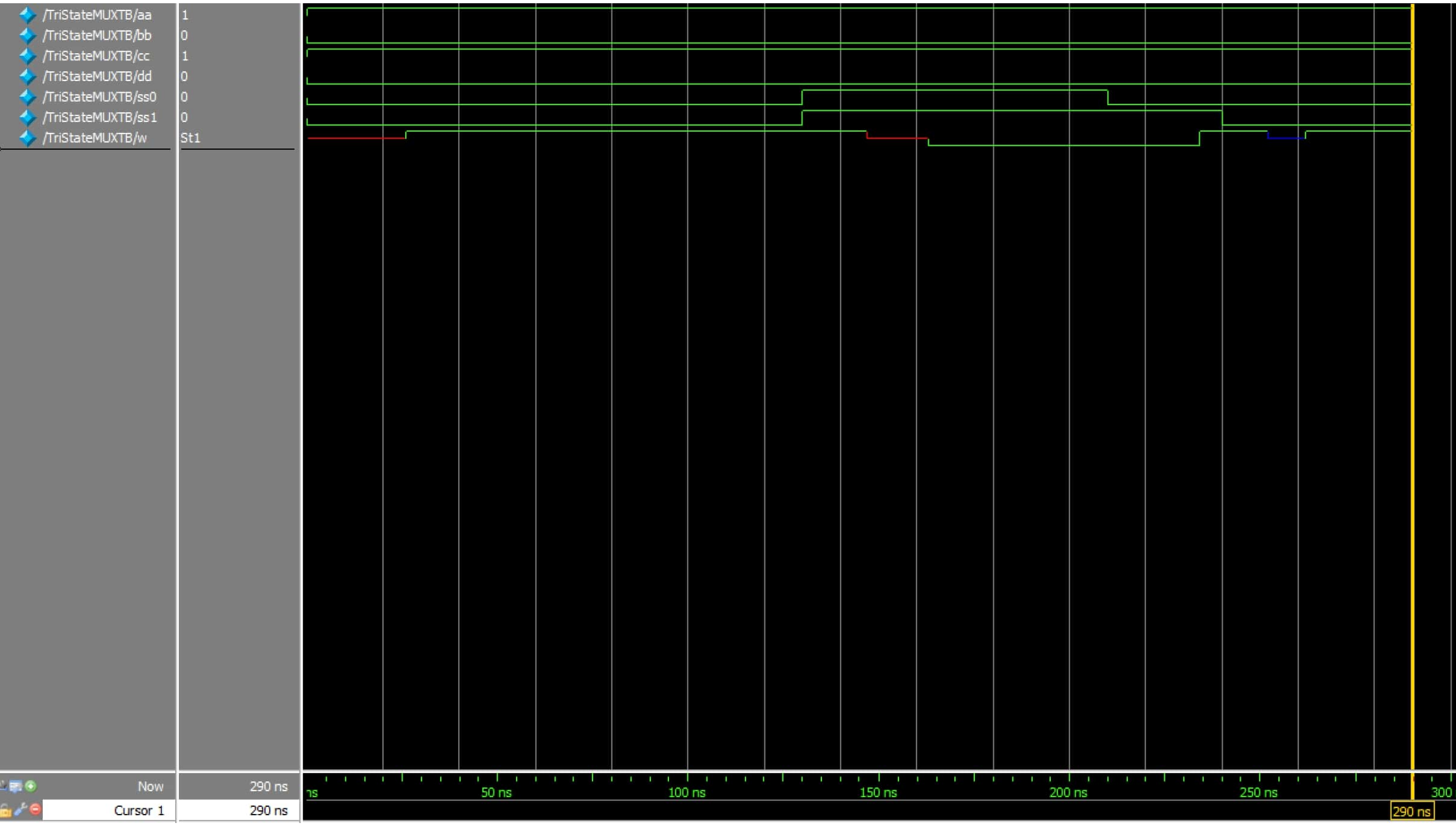
12 ns (Tri_state T01) → Worst_case → 38 ns → T01

Tri-State Buffer 4-To-1 MUX

```
`include "TriStateBuffer.v"

`timescale 1ns/1ns



module TriStateMUX(input a,b,c,d,s0,s1, output w);
    supply1 Vdd;
    supply0 Gnd;

    wire s0_inv, s1_inv;
    wire mux1_out, mux2_out;

    TriStateBuffer invert_s0(.en(Vdd), .a(s0), .w(s0_inv));
    TriStateBuffer invert_s1(.en(Vdd), .a(s1), .w(s1_inv));
    TriStateBuffer select_a(.en(s0_inv), .a(a), .w(mux1_out));
    TriStateBuffer select_b(.en(s0), .a(b), .w(mux1_out));
    TriStateBuffer select_c(.en(s0_inv), .a(c), .w(mux2_out));
    TriStateBuffer select_d(.en(s0), .a(d), .w(mux2_out));

    TriStateBuffer select_mux1(.en(s1_inv), .a(mux1_out), .w(w));
    TriStateBuffer select_mux2(.en(s1), .a(mux2_out), .w(w));

endmodule
```

Tri-State Buffer 4-To-1 MUX testbench

```
`include "TriState4To1MUX.v"

`timescale 1ns/1ns

module TriStateMUXTB();
    reg aa=1,bb=0,cc=1,dd=0,ss0=0,ss1=0;
    wire w;
    TriStateMUX tri_state_mux(aa,bb,cc,dd,ss0,ss1,w);

    initial begin
        #100
        #30 ss0=1;ss1=1;
        #50 bb=0;dd=0;
        #30 ss0=0;
        #30 ss1=0;
        #50 $stop;
    end
endmodule
```



5.

Nand_MUX transistors → 36 → Count

Tri_state MUX transistors → 54 → Count

Power Consumation → Because the number of series transistors in Nand_MUX is higher, its power consumation is also higher

	T0 0	T0 1
Nand_MUX	58 ns	45ns
Tri_state	40ns	38ns

Tri-State buffer MUX vs Nand MUX

```
`include "TriState4To1MUX.v"
`include "4To1MUX.v"

`timescale 1ns/1ns


module TriStateMUXCompareNandMUXTB();
    reg aa=0,bb=0,cc=0,dd=0,s0=0,s1=0;
    wire ww1,ww2;

    NandMUX nand_mux(aa,bb,cc,dd,s0,s1,ww1);
    TriStateMUX tri_state(aa,bb,cc,dd,s0,s1,ww2);

    initial begin
        #50
        #100 s0=1;s1=1;
        #100 bb=1;dd=1;
        #100 aa=1;cc=1;
        #100 $stop;
    end
endmodule
```