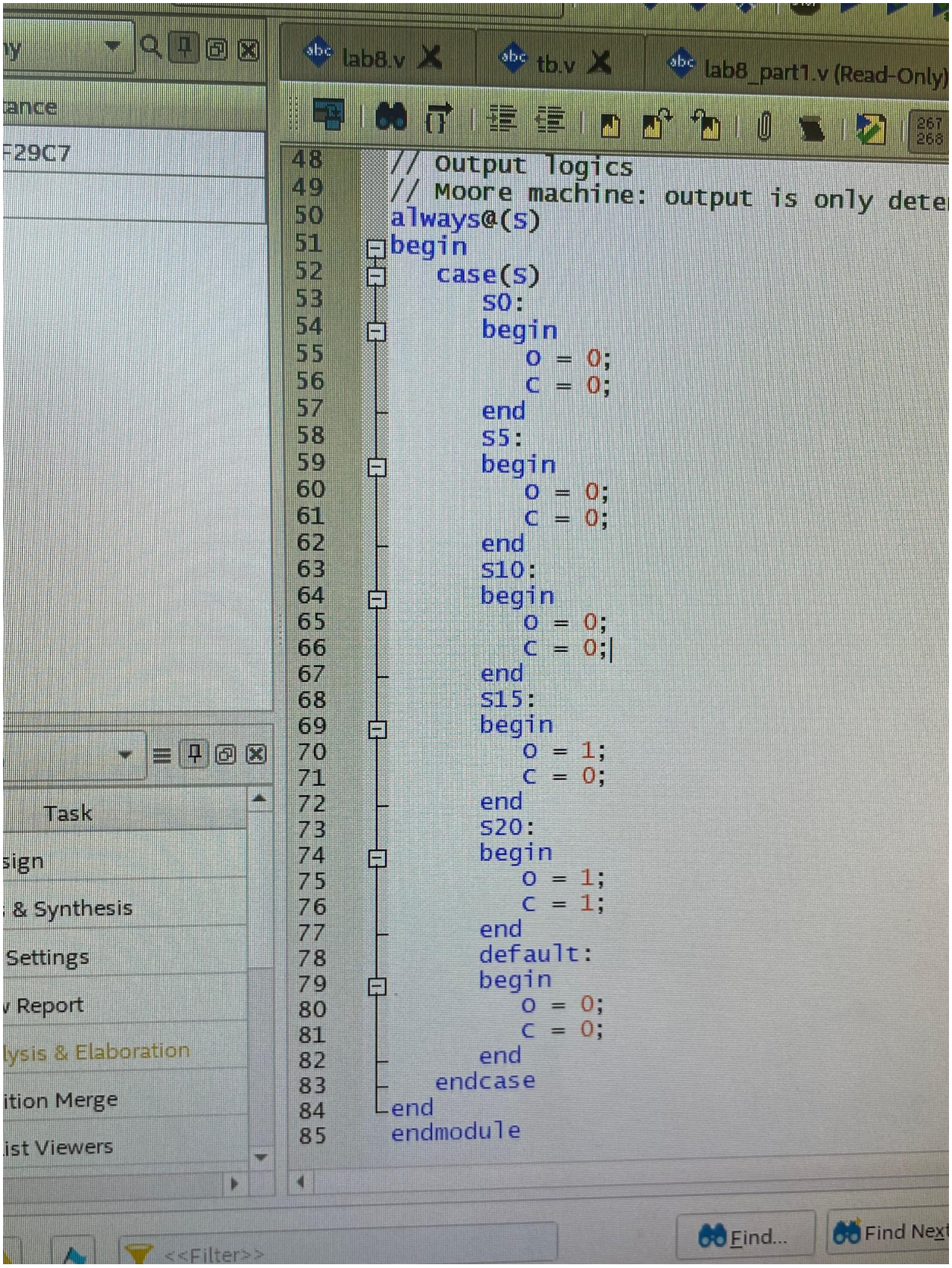
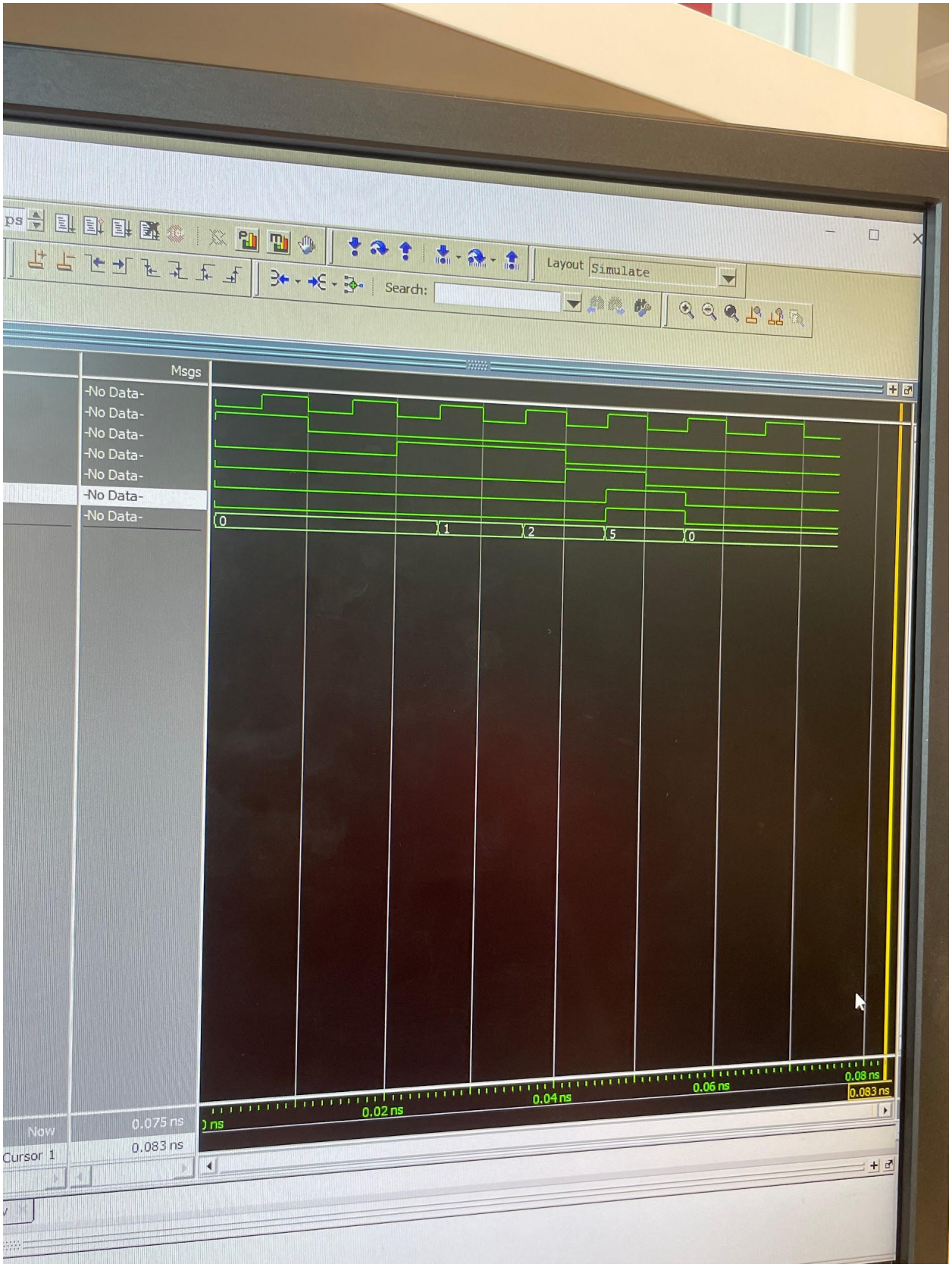
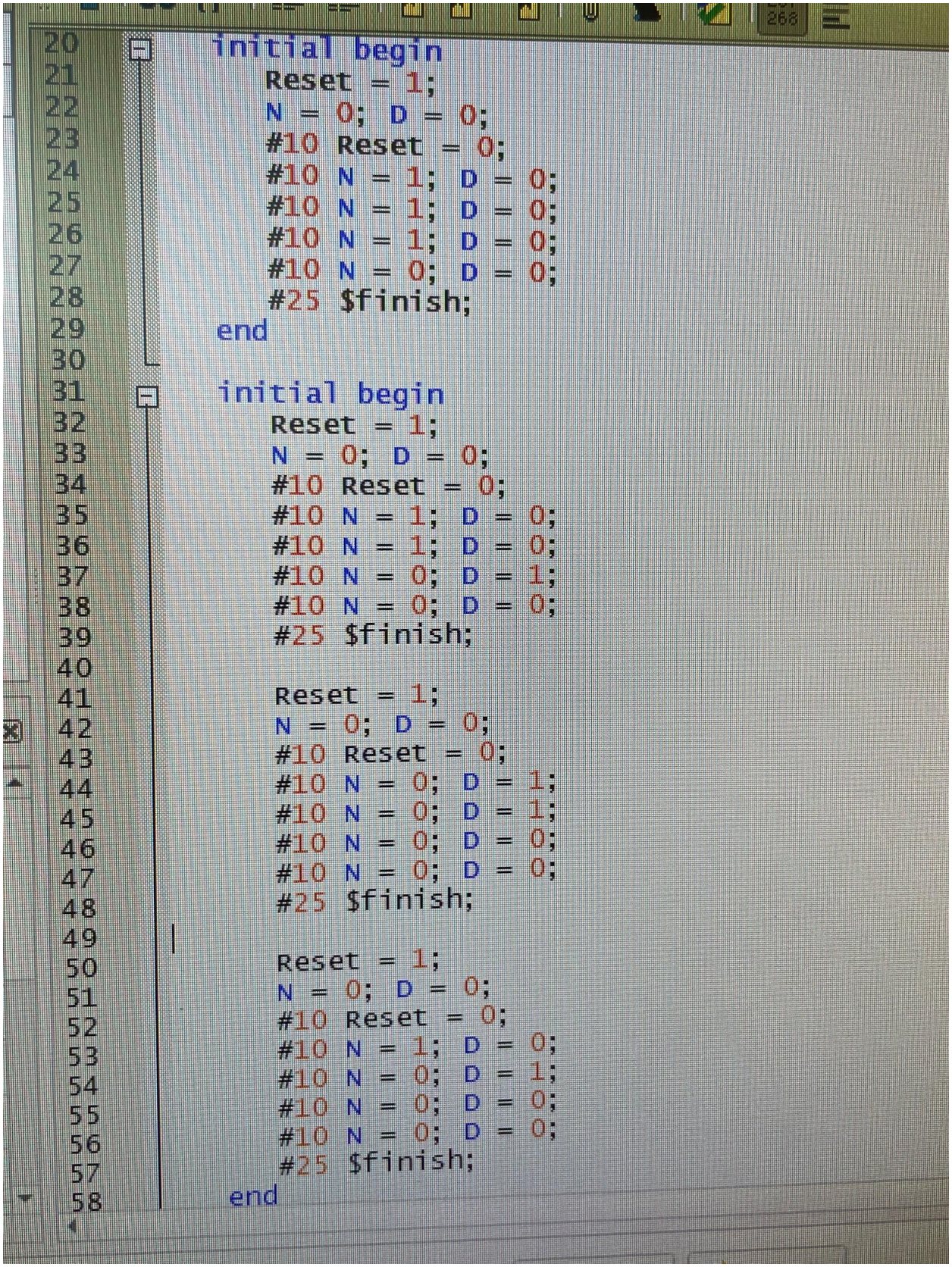
Seana Corners
Ayden Dauenhauer

Lab 8 Post Lab Report

1. My state diagram and state tables were correct in my prelab.
2. I think that the longer Verilog module is more productive because it is more intuitive and easy to understand. Code that is easier to understand and makes more sense at first glance can lead to better understanding from other people. In a team setting, this is important so everyone can contribute and communicate effectively. Although the other version of the code is more concise, it is difficult for others to understand it in an efficient way.
3. Our design would change because we would add more inputs for more coins and bills. The state diagram for instance would completely change because it will have more combinations of possible changes to each state. The coins and bills would operate the same, just at different values. There is a greater chance of getting change (C) because the values of the inputs such as quarter (25) dollars (100, 500, 1000, etc) exceed the change limit of 15.


lab8.v
tb.v
lab8_part1.v (Read-Only)
// Output logics
// Moore machine: output is only dete
always@(s)
begin
case(s)
S0:
begin
o = 0;
c = 0;
end
S5:
begin
o = 0;
c = 0;
end
S10:
begin
o = 0;
c = 0;
end
S15:
begin
o = 1;
c = 0;
end
S20:
begin
o = 1;
c = 1;
end
default:
begin
o = 0;
c = 0;
end
endcase
end
endmodule
Task
sign
& Synthesis
Settings
Report
lysis & Elaboration
ition Merge
ist Viewers
<<Filter>>
Find...
Find Next


Layout
Simulate
Search:
Msgs
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
0
1
2
5
0
0.02 ns
0.04 ns
0.06 ns
0.08 ns
0.083 ns
Now
0.075 ns
Cursor 1
0.083 ns



```
20      initial begin
21          Reset = 1;
22          N = 0; D = 0;
23          #10 Reset = 0;
24          #10 N = 1; D = 0;
25          #10 N = 1; D = 0;
26          #10 N = 1; D = 0;
27          #10 N = 0; D = 0;
28          #25 $finish;
29      end
30
31      initial begin
32          Reset = 1;
33          N = 0; D = 0;
34          #10 Reset = 0;
35          #10 N = 1; D = 0;
36          #10 N = 1; D = 0;
37          #10 N = 0; D = 1;
38          #10 N = 0; D = 0;
39          #25 $finish;
40
41          Reset = 1;
42          N = 0; D = 0;
43          #10 Reset = 0;
44          #10 N = 0; D = 1;
45          #10 N = 0; D = 1;
46          #10 N = 0; D = 0;
47          #10 N = 0; D = 0;
48          #25 $finish;
49
50          Reset = 1;
51          N = 0; D = 0;
52          #10 Reset = 0;
53          #10 N = 1; D = 0;
54          #10 N = 0; D = 1;
55          #10 N = 0; D = 0;
56          #10 N = 0; D = 0;
57          #25 $finish;
58      end
```

b8

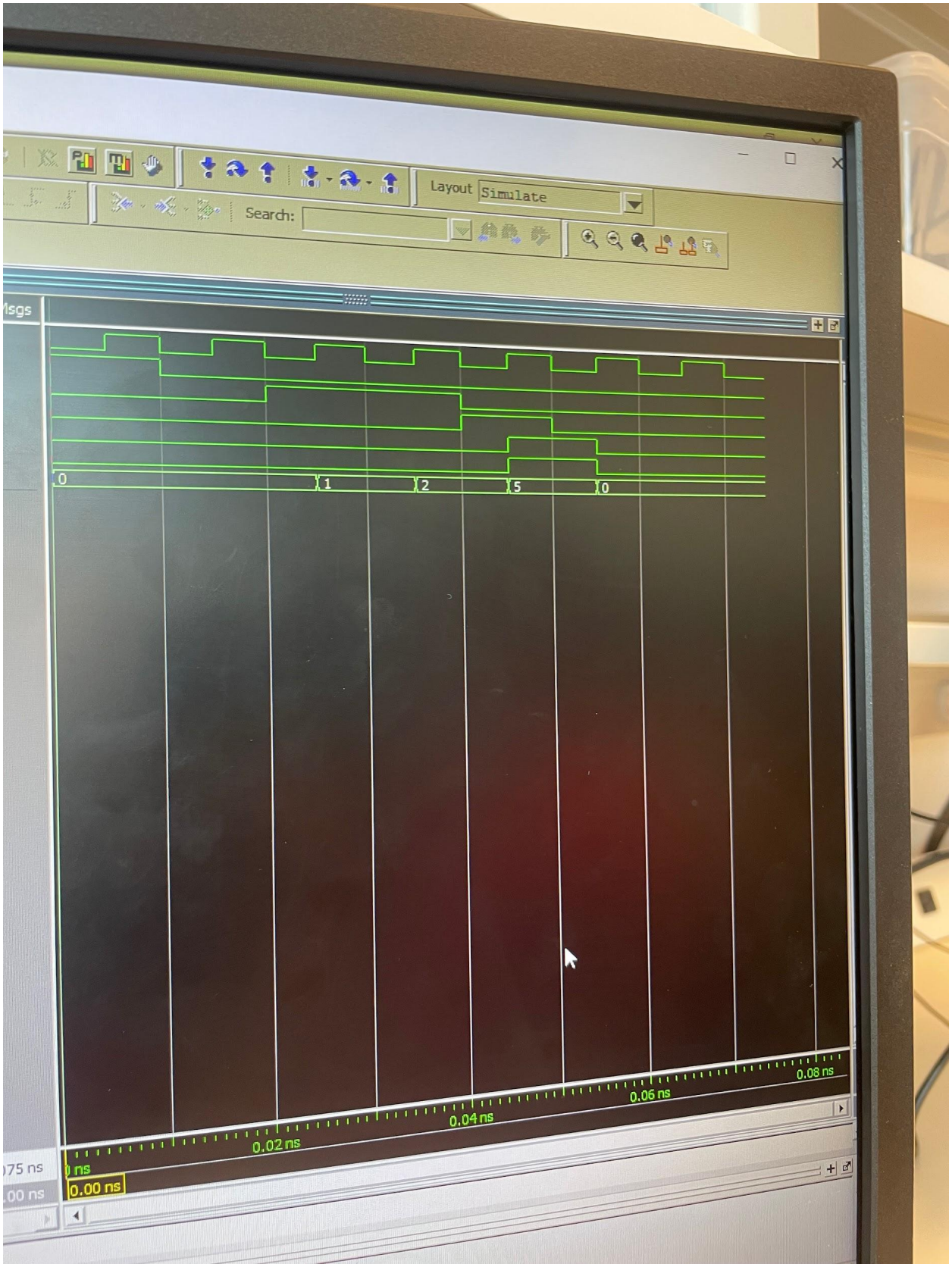
Tools Window Help

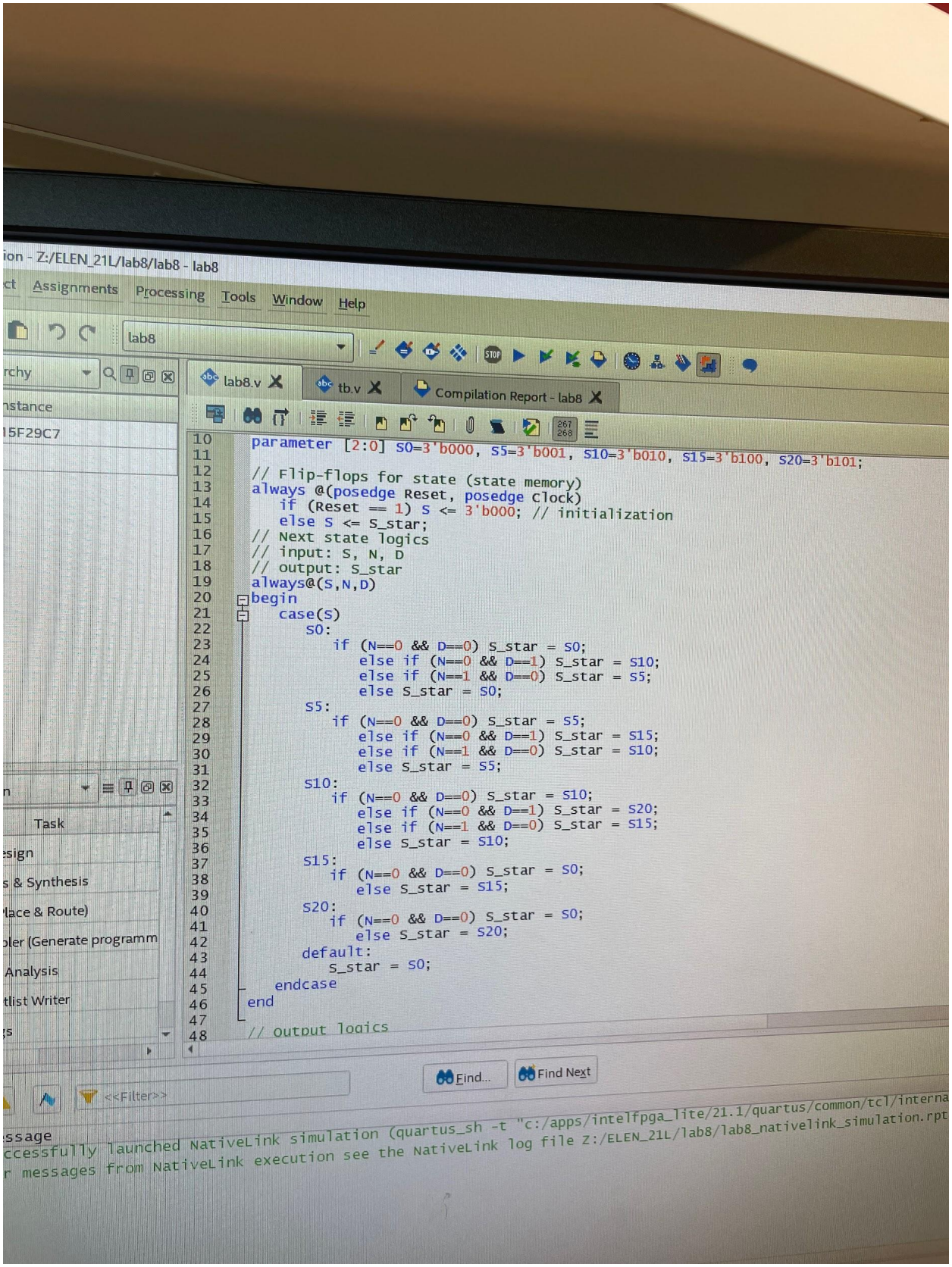
lab8.v | tb.v | lab8_part1.v (Read-Only) | lab8_part2.v (Read-Only) | Compilation Report - lab8

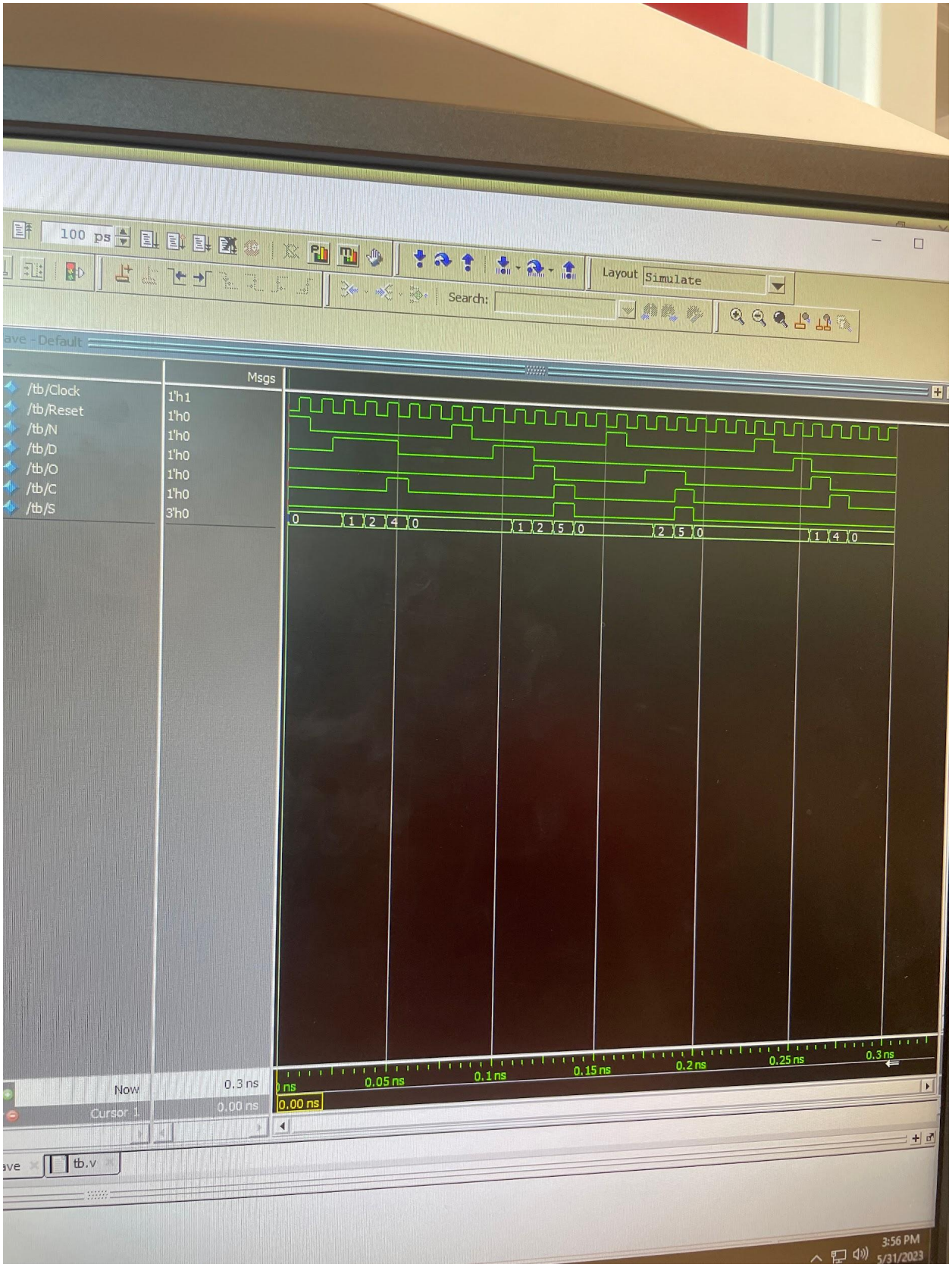
```
module lab8 (Clock, Reset, N, D, O, C, S);

input Clock, Reset; // Clock and Reset
input N, D; // Nickel and Dime sensors (active-high)
output O, C; // Open and Change (active-high)

output reg [2:0] S; // current state
wire [2:0] S_star; // next state

// Flip-flops for state (state memory)
always @(posedge Reset, posedge Clock)
    if (Reset == 1) S <= 3'b000; // initialization
    else S <= S_star;

// Next state logics
// input: S, N, D
// output: S_star
assign S_star[2] = ~S[2] & (S[0] & D | S[1] & D | S[1] & N);
assign S_star[1] = ~S[2] & (~S[1] & ~S[0] & D | S[0] & N | S[1] & ~N & ~D);
assign S_star[0] = ~S[2] & (~S[1] & ~S[0] & N | S[1] & D | S[0] & ~N & ~D);

// Output logics
// Moore machine: output is only determined by the current state (S), that is, input is S
assign O = S[2];
assign C = S[2] & S[0];

endmodule
```

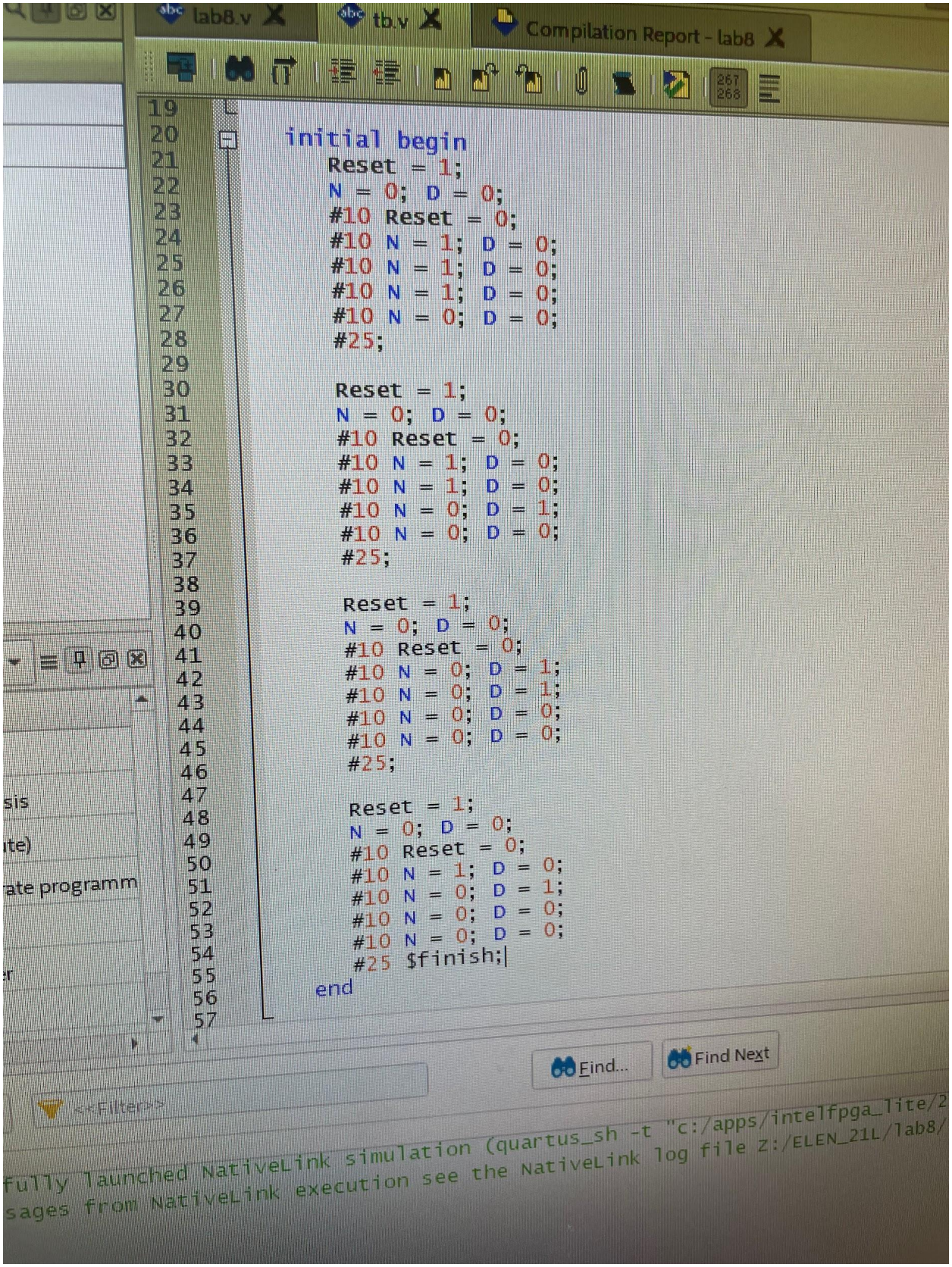
Find... Find Next

```
NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/internal/nativel
iveLink execution see the NativeLink log file Z:/ELEN_21L/lab8/lab8_nativelink_simulation.rpt
NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/internal/nativeli
iveLink execution see the NativeLink log file Z:/ELEN_21L/lab8/lab8_nativelink_simulation.rpt
NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/internal/nativeli
iveLink execution see the NativeLink log file Z:/ELEN_21L/lab8/lab8_nativelink_simulation.rpt
NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/internal/nativeli
iveLink execution see the NativeLink log file Z:/ELEN_21L/lab8/lab8_nativelink_simulation.rpt
```


Layout
Simulate
Search:
Msgs
0
1
2
5
0
0.02 ns
0.04 ns
0.06 ns
0.08 ns
0.00 ns


ion - Z:/ELEN_21L/lab8/lab8 - lab8
Assignments
Processing
Tools
Window
Help
lab8
lab8.v
tb.v
Compilation Report - lab8
parameter [2:0] S0=3'b000, S5=3'b001, S10=3'b010, S15=3'b100, S20=3'b101;
// Flip-flops for state (state memory)
always @(posedge Reset, posedge clock)
if (Reset == 1) S <= 3'b000; // initialization
else S <= S_star;
// Next state logics
// input: S, N, D
// output: S_star
always@(S,N,D)
begin
case(S)
S0:
if (N==0 && D==0) S_star = S0;
else if (N==0 && D==1) S_star = S10;
else if (N==1 && D==0) S_star = S5;
else S_star = S0;
S5:
if (N==0 && D==0) S_star = S5;
else if (N==0 && D==1) S_star = S15;
else if (N==1 && D==0) S_star = S10;
else S_star = S5;
S10:
if (N==0 && D==0) S_star = S10;
else if (N==0 && D==1) S_star = S20;
else if (N==1 && D==0) S_star = S15;
else S_star = S10;
S15:
if (N==0 && D==0) S_star = S0;
else S_star = S15;
S20:
if (N==0 && D==0) S_star = S0;
else S_star = S20;
default:
S_star = S0;
endcase
end
// Output logics
Task
Find...
Find Next
<<Filter>>
Successfully launched NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/interna
Successfully launched NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/21.1/quartus/common/tcl/interna
For messages from NativeLink execution see the NativeLink log file Z:/ELEN_21L/lab8/lab8_nativelink_simulation.rpt


100 ps
Layout
Simulate
Search:
Msgs
/tb/Clock
/tb/Reset
/tb/N
/tb/D
/tb/O
/tb/C
/tb/S
1'h1
1'h0
1'h0
1'h0
1'h0
1'h0
3'h0
0
1
2
4
0
1
2
5
0
2
5
0
1
4
0
Now
0.3 ns
Cursor 1
0.00 ns
0 ns
0.05 ns
0.1 ns
0.15 ns
0.2 ns
0.25 ns
0.3 ns
0.00 ns
tb.v
3:56 PM
5/31/2023


lab8.v
tb.v
Compilation Report - lab8
initial begin
Reset = 1;
N = 0; D = 0;
#10 Reset = 0;
#10 N = 1; D = 0;
#10 N = 1; D = 0;
#10 N = 1; D = 0;
#10 N = 0; D = 0;
#25;
Reset = 1;
N = 0; D = 0;
#10 Reset = 0;
#10 N = 1; D = 0;
#10 N = 1; D = 0;
#10 N = 0; D = 1;
#10 N = 0; D = 0;
#25;
Reset = 1;
N = 0; D = 0;
#10 Reset = 0;
#10 N = 0; D = 1;
#10 N = 0; D = 1;
#10 N = 0; D = 0;
#10 N = 0; D = 0;
#25;
Reset = 1;
N = 0; D = 0;
#10 Reset = 0;
#10 N = 1; D = 0;
#10 N = 0; D = 1;
#10 N = 0; D = 0;
#10 N = 0; D = 0;
#25 $finish;
end
Find...
Find Next
<<Filter>>
fully launched NativeLink simulation (quartus_sh -t "c:/apps/intelfpga_lite/2
sages from NativeLink execution see the NativeLink log file Z:/ELEN_211/lab8/

Verilog:

```
module lab8 (Clock, Reset, N, D, O, C, S);

input Clock, Reset; // Clock and Reset
input N, D; // Nickel and Dime sensors (active-high)
output O, C; // Open and Change (active-high)

output reg [2:0] S; // current state
wire [2:0] S_star; // next state

// Flip-flops for state (state memory)
always @(posedge Reset, posedge Clock)
	if (Reset == 1) S <= 3'b000; // initialization
	else S <= S_star;

// Next state logics
// input: S, N, D
// output: S_star
assign S_star[2] = ~S[2] & (S[0] & D | S[1] & D | S[1] & N);
assign S_star[1] = ~S[2] & (~S[1] & ~S[0] & D | S[0] & N | S[1] & ~N & ~D);
assign S_star[0] = ~S[2] & (~S[1] & ~S[0] & N | S[1] & D | S[0] & ~N & ~D);

// Output logics
// Moore machine: output is only determined by the current state (S), that is, input is S
assign O = S[2];
assign C = S[2] & S[0];

endmodule
```

```
module lab8 (Clock, Reset, N, D, O, C, S);

input Clock, Reset; // Clock and Reset
input N, D; // Nickel and Dime sensors (active-high)
output reg O, C; // Open and Change (active-high)

output reg [2:0] S; // current state
reg [2:0] S_star; // next state

parameter [2:0] S0=3'b000, S5=3'b001, S10=3'b010, S15=3'b100, S20=3'b101;
```

```
// Flip-flops for state (state memory)
always @(posedge Reset, posedge Clock)
	if (Reset == 1) S <= 3'b000; // initialization
	else S <= S_star;
// Next state logics
// input: S, N, D
// output: S_star
always@(S,N,D)
begin
	case(S)
		S0:
			if (N==0 && D==0) S_star = S0;
				else if (N==0 && D==1) S_star = S10;
				else if (N==1 && D==0) S_star = S5;
				else S_star = S0;
		S5:
			if (N==0 && D==0) S_star = S5;
				else if (N==0 && D==1) S_star = S15;
				else if (N==1 && D==0) S_star = S10;
				else S_star = S5;
		S10:
			if (N==0 && D==0) S_star = S10;
				else if (N==0 && D==1) S_star = S20;
				else if (N==1 && D==0) S_star = S15;
				else S_star = S10;
		S15:
			if (N==0 && D==0) S_star = S0;
				else S_star = S15;
		S20:
			if (N==0 && D==0) S_star = S0;
				else S_star = S20;
		default:
			S_star = S0;
	endcase
end

// Output logics
// Moore machine: output is only determined by the current state (S), that is, input is S
always@(S)
begin
	case(S)
		S0:
		begin
			O = 0;
```

```
                C = 0;
            end
            S5:
            begin
                O = 0;
                C = 0;
            end
            S10:
            begin
                O = 0;
                C = 0;
            end
            S15:
            begin
                O = 1;
                C = 0;
            end
            S20:
            begin
                O = 1;
                C = 1;
            end
            default:
            begin
                O = 0;
                C = 0;
            end
        endcase
end
endmodule
```