22bec121

Experiment – 8 Date:-14/3/2024

Lab Work

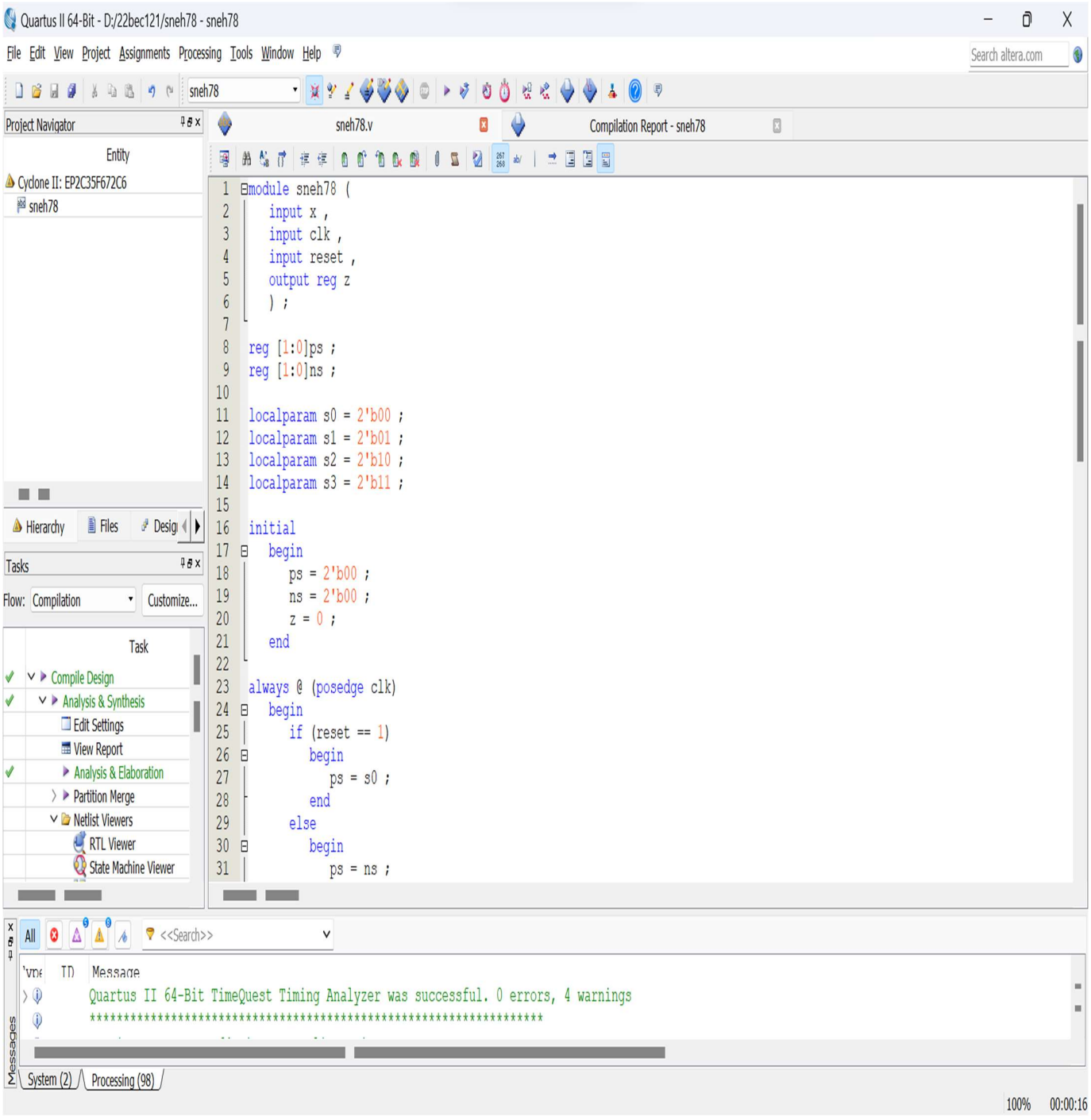
Q1 .Design a Mealy Based FSM 101 Overlapping Sequence Detector .

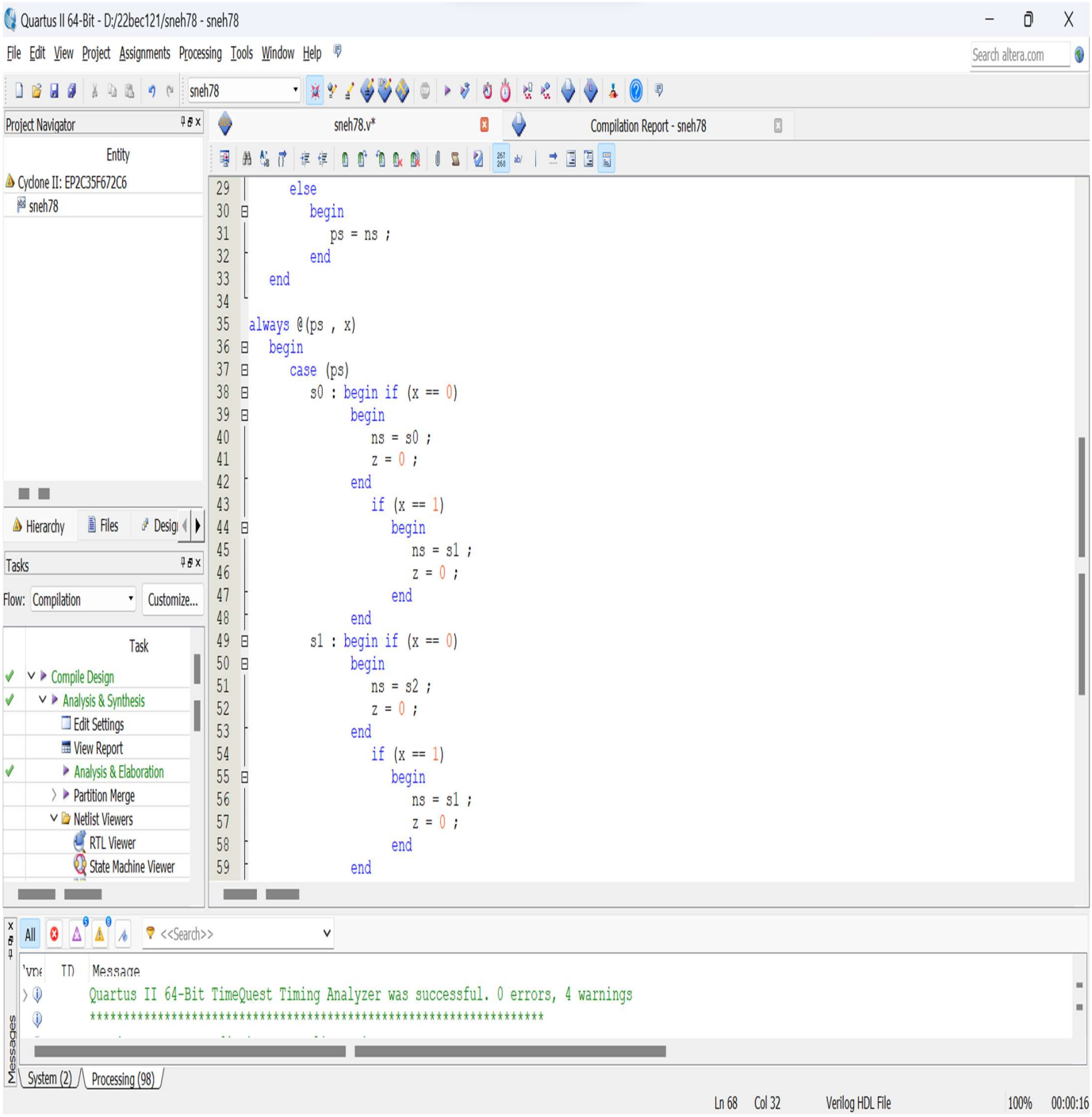
Code (Mealy FSM)

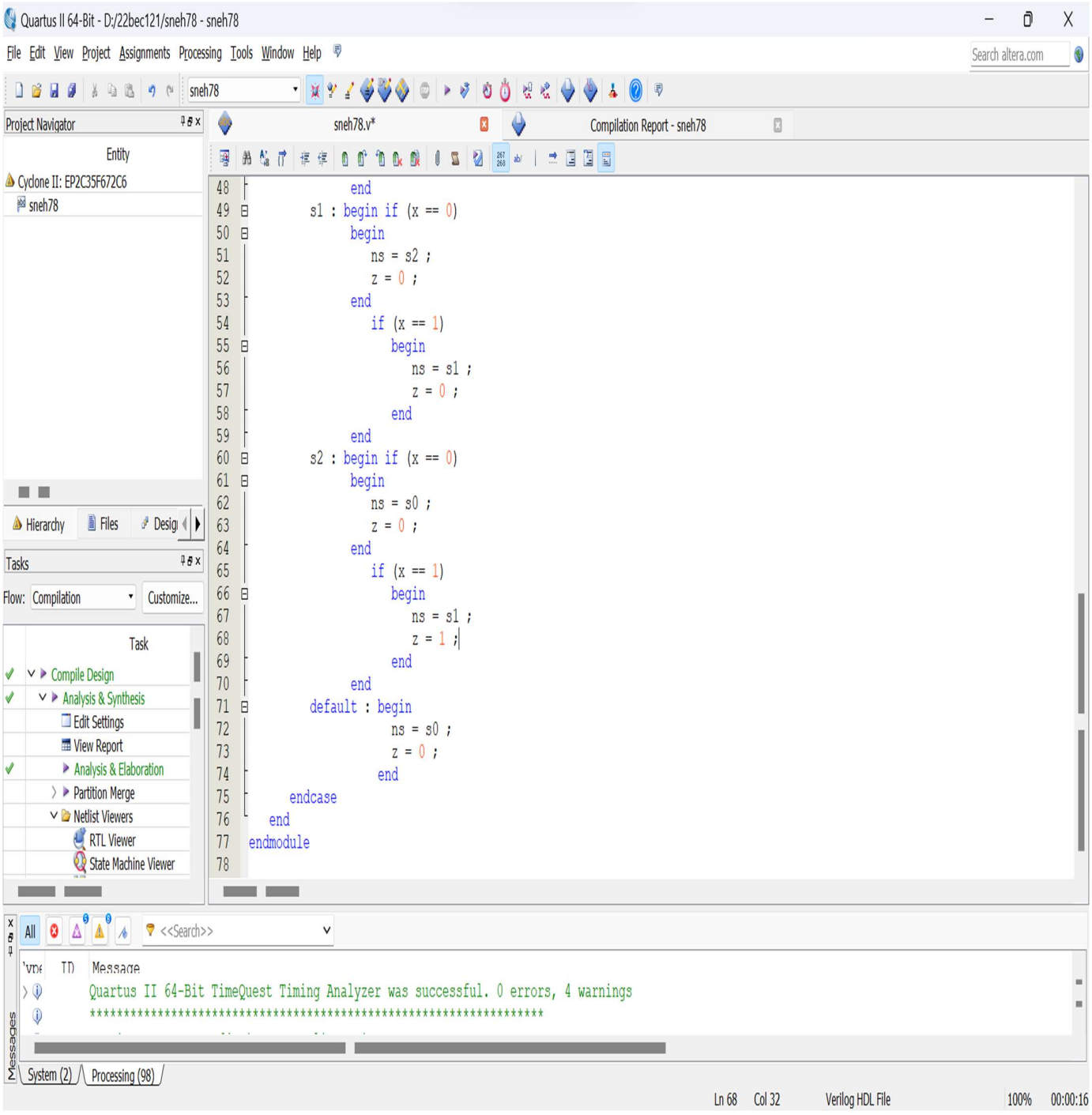
```
module sneh78 (
        input x ,
        input clk ,
        input reset ,
        output reg z
        ) ;

reg [1:0]ps ;
reg [1:0]ns ;

localparam s0 = 2'b00 ;
localparam s1 = 2'b01 ;
localparam s2 = 2'b10 ;
localparam s3 = 2'b11 ;

initial
        begin
                ps = 2'b00 ;
                ns = 2'b00 ;
                z = 0 ;
        end

always @ (posedge clk)
        begin
```

```
        if (reset == 1)
                begin
                        ps = s0 ;
                end
        else
                begin
                        ps = ns ;
                end
    end

always @(ps , x)
    begin
        case (ps)
            s0 : begin if (x == 0)
                    begin
                        ns = s0 ;
                        z = 0 ;
                    end
                        if (x == 1)
                            begin
                                ns = s1 ;
                                z = 0 ;
                            end
                    end
            s1 : begin if (x == 0)
                    begin
                        ns = s2 ;
                        z = 0 ;
                    end
                        if (x == 1)
                            begin
                                ns = s1 ;
                                z = 0 ;
```

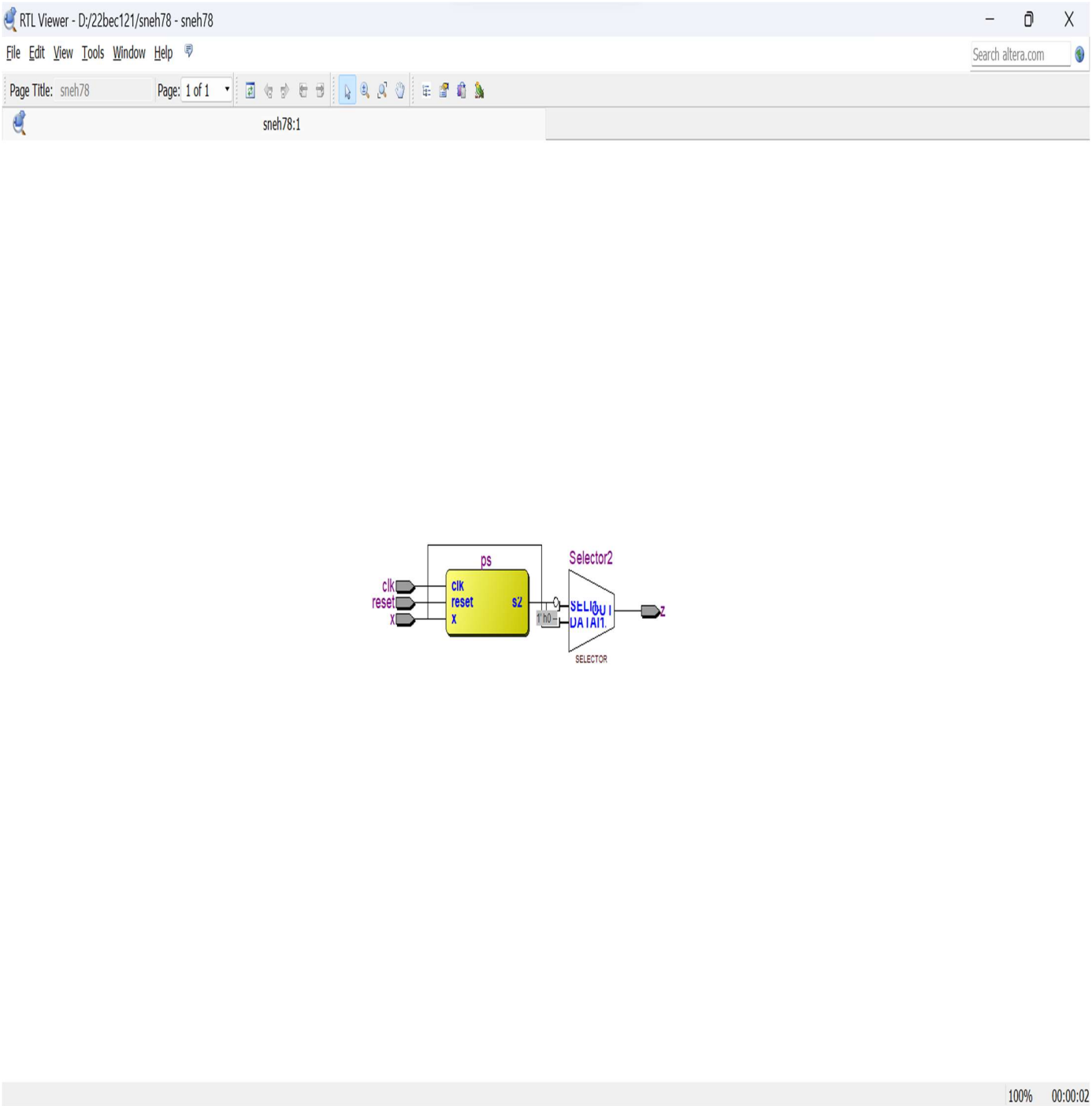
```
                                        end
                        end
                s2 : begin if (x == 0)
                        begin
                                ns = s0 ;
                                z = 0 ;
                        end
                                if (x == 1)
                                        begin
                                                ns = s1 ;
                                                z = 1 ;
                                        end
                        end
                default : begin
                                        ns = s0 ;
                                        z = 0 ;
                                end
        endcase
    end
endmodule
```


Quartus II 64-Bit - D:/22bec121/sneh78 - sneh78
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh78
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh78
Hierarchy
Files
Desig
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Netlist Viewers
RTL Viewer
State Machine Viewer
sneh78.v
Compilation Report - sneh78
1 module sneh78 (
2 input x ,
3 input clk ,
4 input reset ,
5 output reg z
6) ;
7
8 reg [1:0]ps ;
9 reg [1:0]ns ;
10
11 localparam s0 = 2'b00 ;
12 localparam s1 = 2'b01 ;
13 localparam s2 = 2'b10 ;
14 localparam s3 = 2'b11 ;
15
16 initial
17 begin
18 ps = 2'b00 ;
19 ns = 2'b00 ;
20 z = 0 ;
21 end
22
23 always @ (posedge clk)
24 begin
25 if (reset == 1)
26 begin
27 ps = s0 ;
28 end
29 else
30 begin
31 ps = ns ;
All
<<Search>>
Type ID Message
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
****
Messages
System (2)
Processing (98)
100%
00:00:16

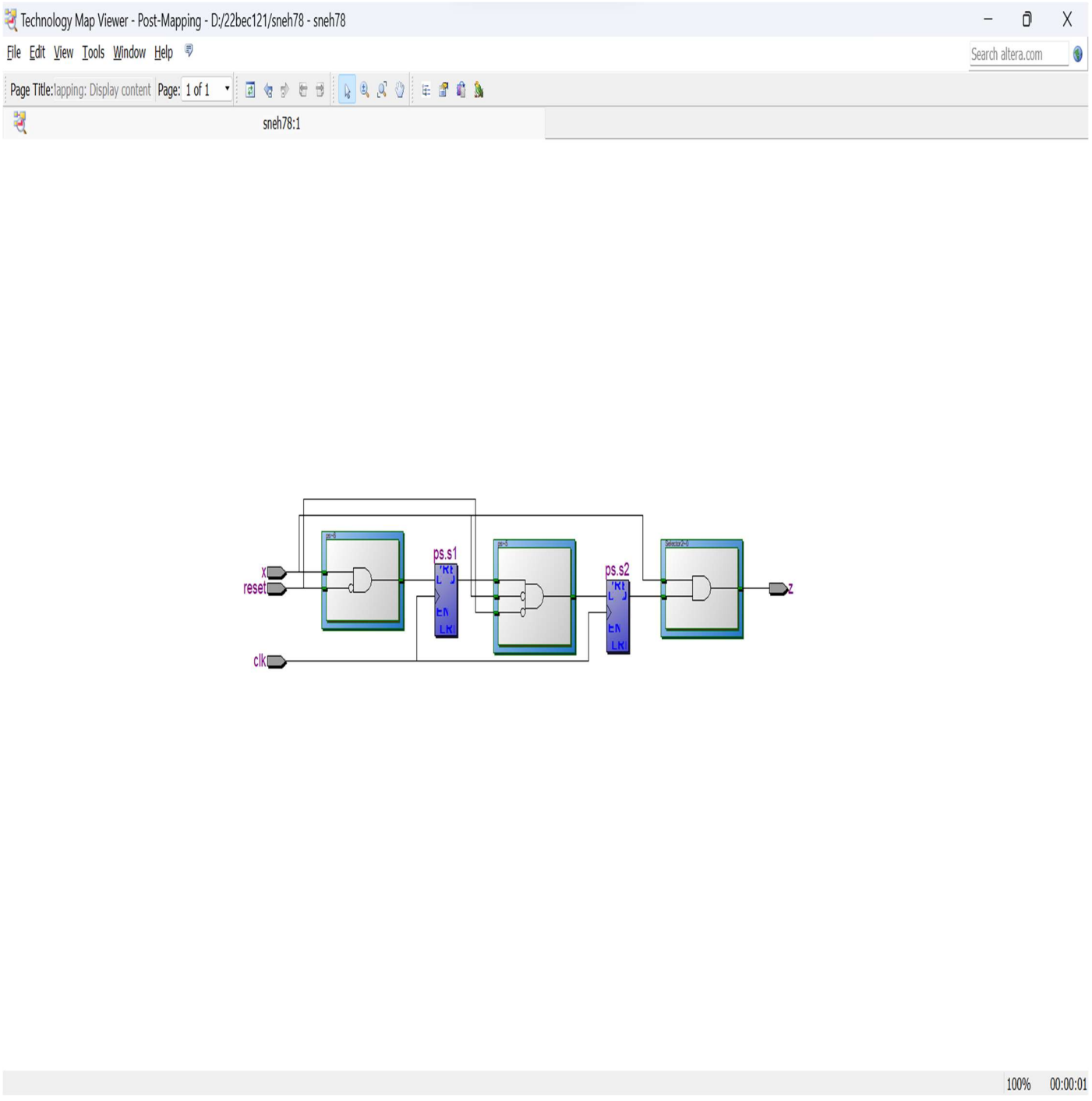

Quartus II 64-Bit - D:/22bec121/sneh78 - sneh78
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh78
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh78
sneh78.v*
Compilation Report - sneh78
else
begin
ps = ns ;
end
end
always @(ps , x)
begin
case (ps)
s0 : begin if (x == 0)
begin
ns = s0 ;
z = 0 ;
end
if (x == 1)
begin
ns = s1 ;
z = 0 ;
end
end
s1 : begin if (x == 0)
begin
ns = s2 ;
z = 0 ;
end
if (x == 1)
begin
ns = s1 ;
z = 0 ;
end
end
Hierarchy
Files
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Netlist Viewers
RTL Viewer
State Machine Viewer
All
<<Search>>
Type
ID
Message
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
System (2)
Processing (98)
Ln 68
Col 32
Verilog HDL File
100%
00:00:16


Quartus II 64-Bit - D:/22bec121/sneh78 - sneh78
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh78
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh78
sneh78.v*
Compilation Report - sneh78
Hierarchy
Files
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Netlist Viewers
RTL Viewer
State Machine Viewer
48 end
49 s1 : begin if (x == 0)
50 begin
51 ns = s2 ;
52 z = 0 ;
53 end
54 if (x == 1)
55 begin
56 ns = s1 ;
57 z = 0 ;
58 end
59 end
60 s2 : begin if (x == 0)
61 begin
62 ns = s0 ;
63 z = 0 ;
64 end
65 if (x == 1)
66 begin
67 ns = s1 ;
68 z = 1 ;
69 end
70 end
71 default : begin
72 ns = s0 ;
73 z = 0 ;
74 end
75 endcase
76 end
77 endmodule
78
All
<<Search>>
Type ID Message
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
System (2)
Processing (98)
Ln 68 Col 32 Verilog HDL File 100% 00:00:16

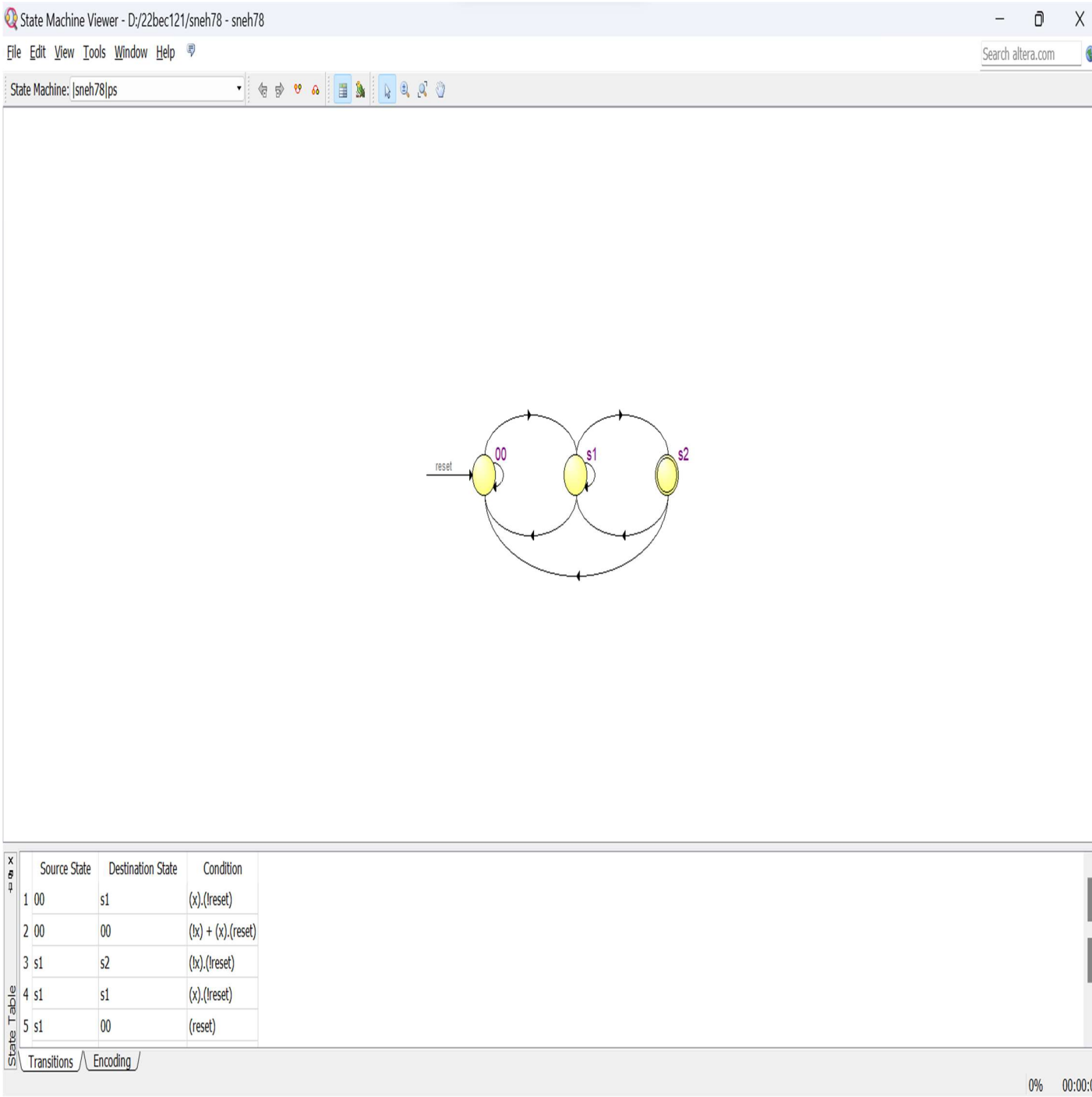
Output 1)Cyclone II (RTL View)



2)Cyclone II (TTL View)

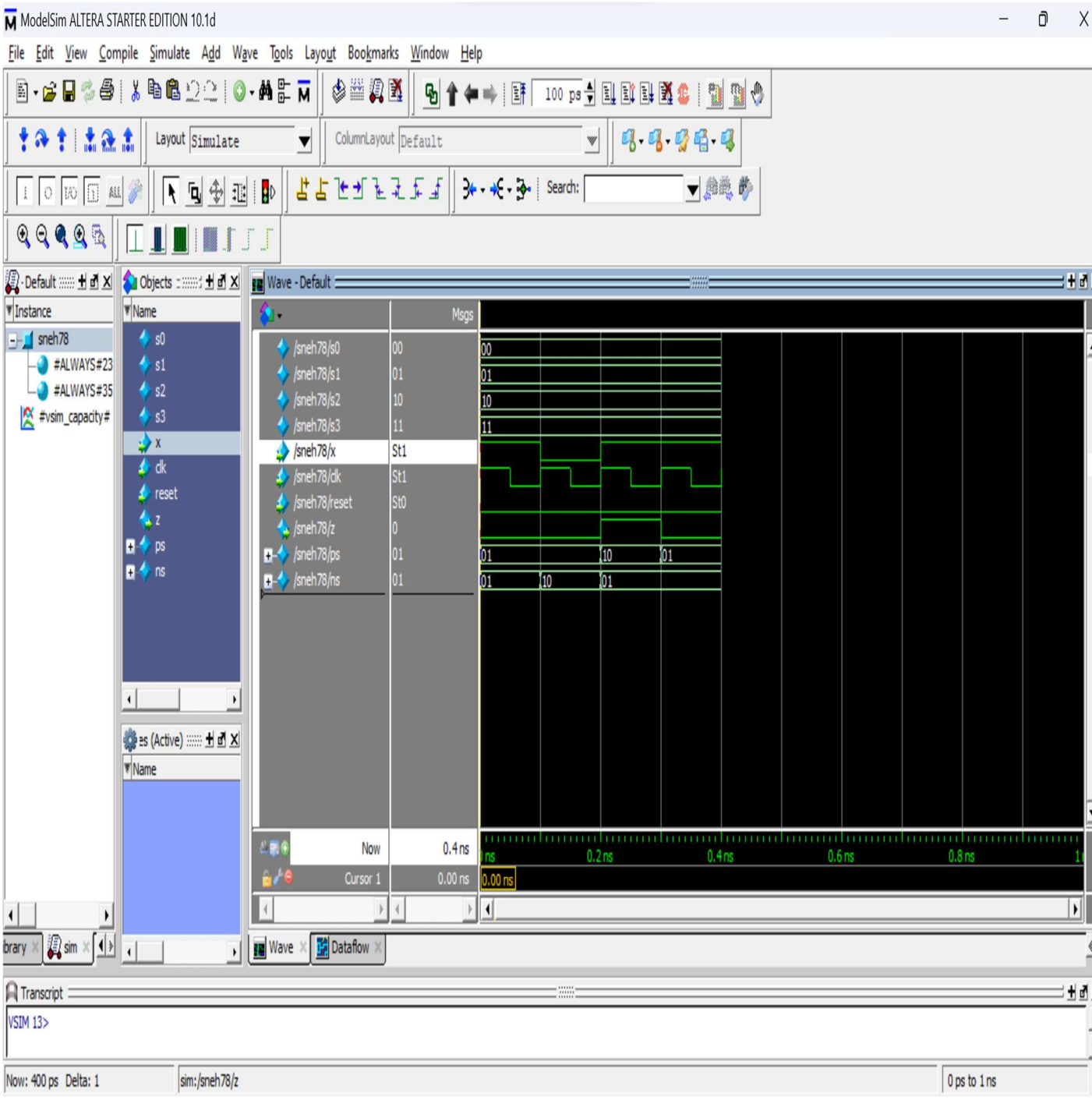


3) State Diagram

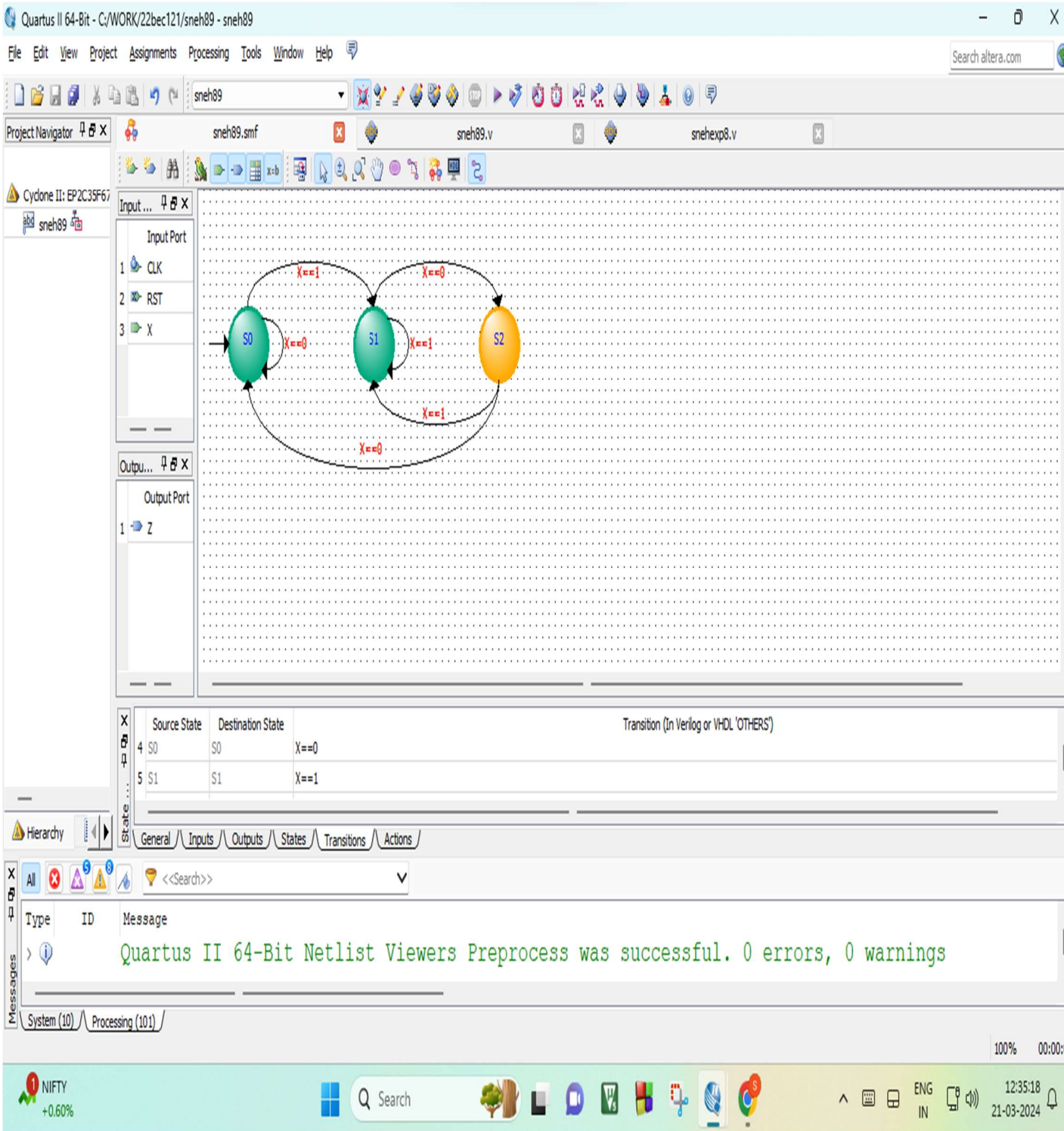


	Source State	Destination State	Condition
1	00	s1	(x).(!reset)
2	00	00	(!x) + (x).(reset)
3	s1	s2	(!x).(!reset)
4	s1	s1	(x).(!reset)
5	s1	00	(reset)

Simulation Results :-



State Machine Wizard



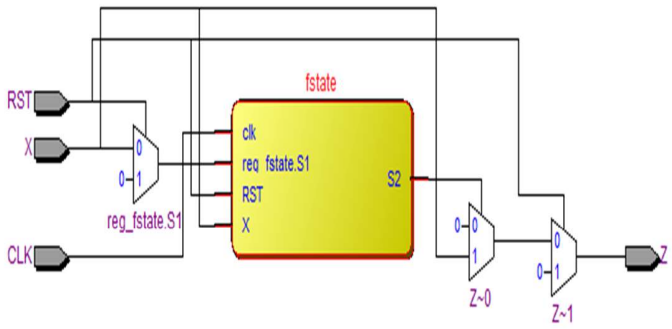
Generated Code

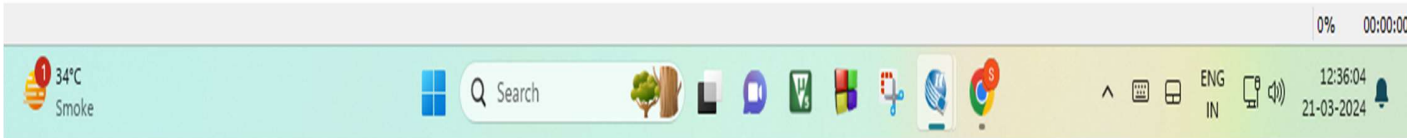
```
// Copyright (C) 1991-2013 Altera Corporation
// Your use of Altera Corporation's design tools, logic functions
// and other software and tools, and its AMPP partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Altera Program License
// Subscription Agreement, Altera MegaCore Function License
// Agreement, or other applicable license agreement, including,
// without limitation, that your use is for the sole purpose of
// programming logic devices manufactured by Altera and sold by
// Altera or its authorized distributors.  Please refer to the
// applicable agreement for further details.

// Generated by Quartus II Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
// Created on Thu Mar 21 12:00:21 2024

// synthesis message_off 10175

`timescale 1ns/1ns

module sneh89 (
    CLK,RST,X,
    Z);

    input CLK;
    input RST;
    input X;
    tri0 RST;
    tri0 X;
    output Z;
    reg Z;
    reg [2:0] fstate;
    reg [2:0] reg_fstate;
    parameter S0=0,S1=1,S2=2;
```

```
always @(posedge CLK)
begin
    if (CLK) begin
        fstate <= reg_fstate;
    end
end

always @(fstate or X)
begin
    if (RST) begin
        reg_fstate <= S0;
        Z <= 1'b0;
    end
    else begin
        Z <= 1'b0;
        case (fstate)
            S0: begin
                if ((X == 1'b1))
                    reg_fstate <= S1;
                else if ((X == 1'b0))
                    reg_fstate <= S0;
                // Inserting 'else' block to prevent latch inference
                else
                    reg_fstate <= S0;

                Z <= 1'b0;
            end
            S1: begin
                if ((X == 1'b0))
                    reg_fstate <= S2;
                else if ((X == 1'b1))
                    reg_fstate <= S1;
                // Inserting 'else' block to prevent latch inference
                else
                    reg_fstate <= S1;

                Z <= 1'b0;
            end
            S2: begin
                if ((X == 1'b0))
```

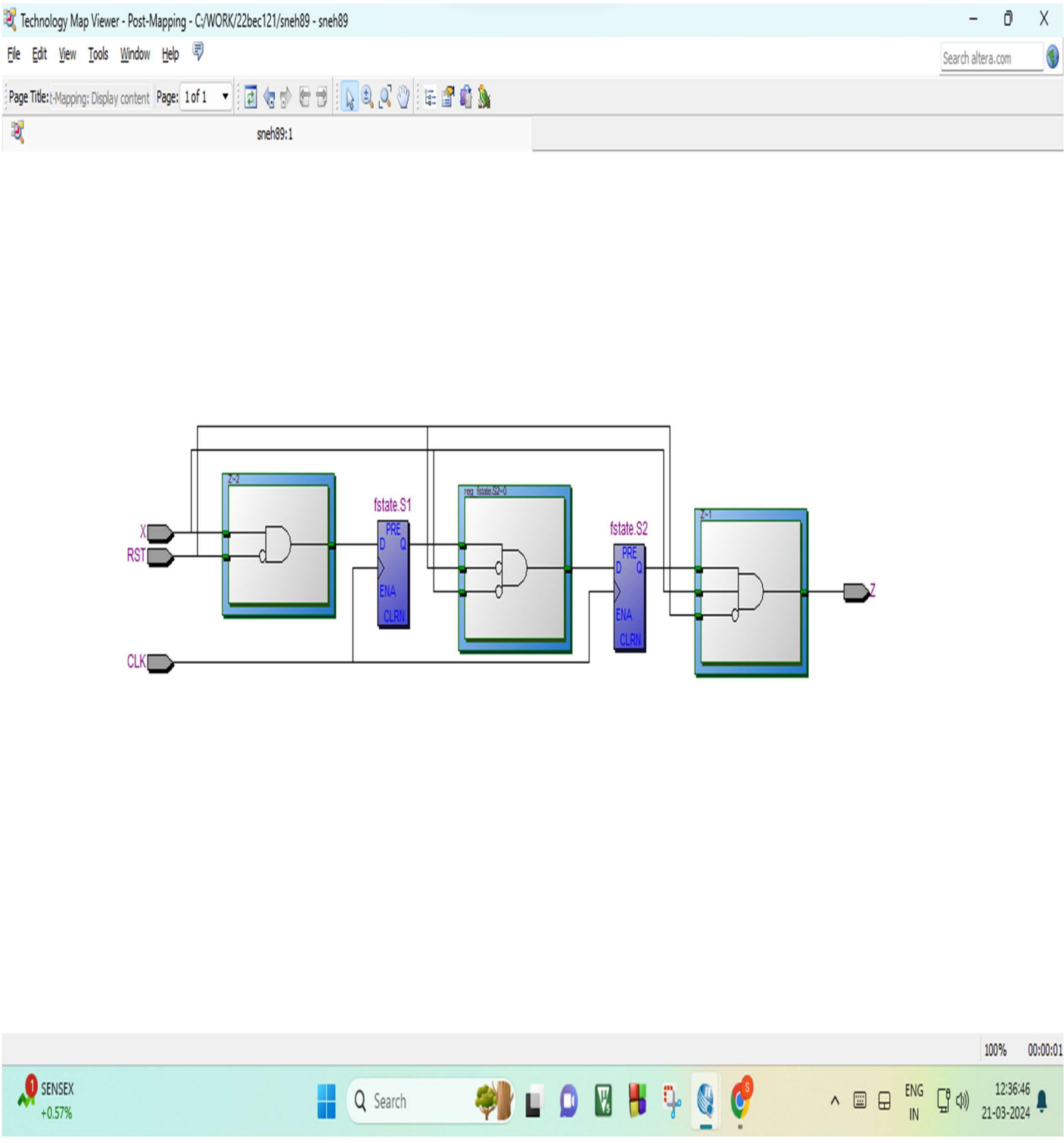
```
                reg_fstate <= S0;
            else if ((X == 1'b1))
                reg_fstate <= S1;
            // Inserting 'else' block to prevent latch inference
            else
                reg_fstate <= S2;

            if ((X == 1'b1))
                Z <= 1'b1;
            // Inserting 'else' block to prevent latch inference
            else
                Z <= 1'b0;
          end
          default: begin
            Z <= 1'bx;
            $display ("Reach undefined state");
          end
        endcase
      end
    end
endmodule // sneh89
```

Output 1)Cyclone II (RTL View)

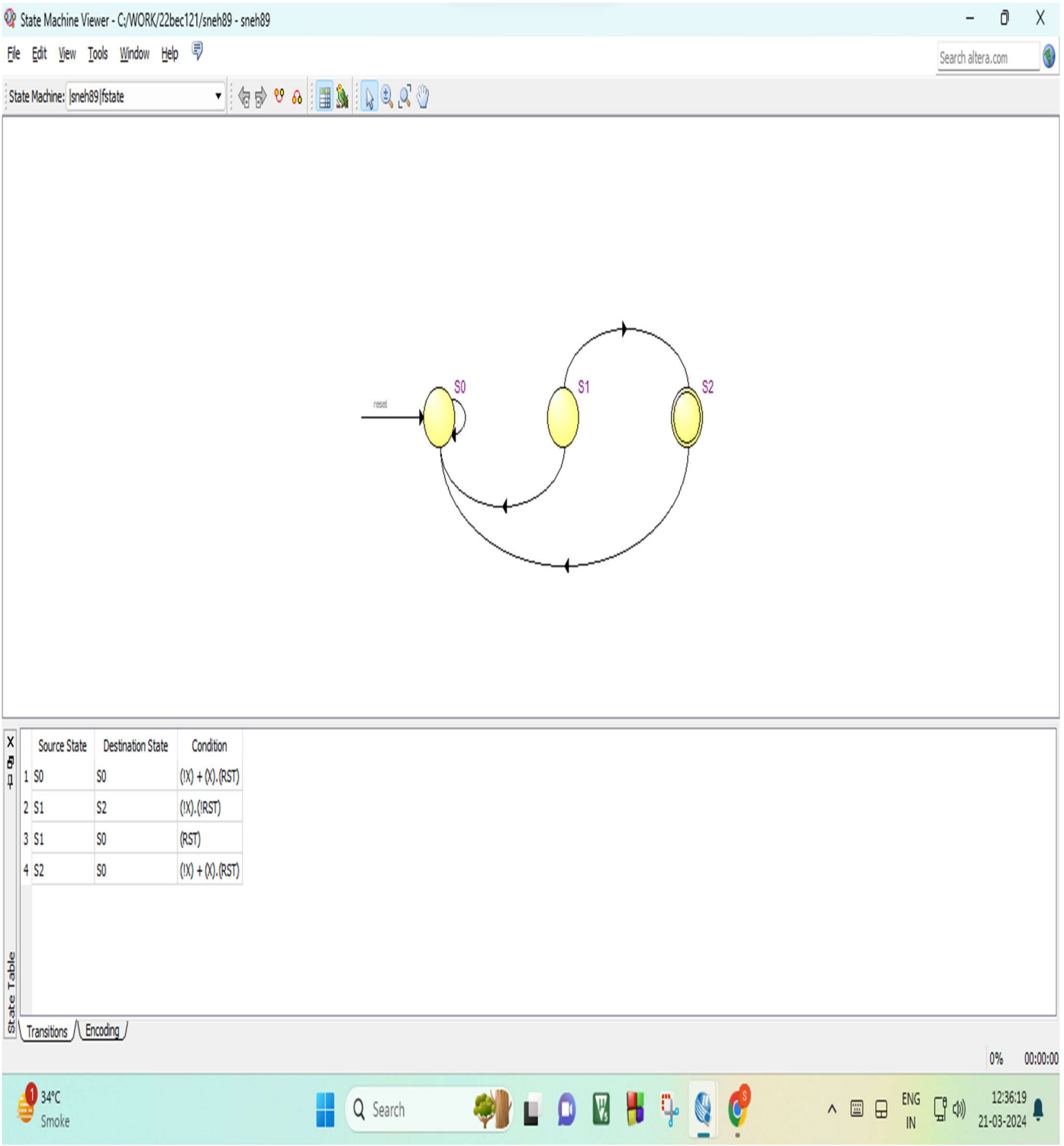




2)Cyclone II (TTL View)



3)State Machine View



	Source State	Destination State	Condition
1	S0	S0	(!X) + (X).(RST)
2	S1	S2	(!X).(!RST)
3	S1	S0	(RST)
4	S2	S0	(!X) + (X).(RST)

Conclusion :- In this experiment we learnt to design a Mealy FSM in which output depends on Present State and Inputs .

We also viewed how the software designs a State Diagram based on the code .

We also learnt how RTL and TTL forms for a Mealy FSM 101 Sequence Detector . We also generated Verilog code from State Diagram .

We simulated the FSM on ModelSim software to view its functioning . We also implemented Mealy FSM Sequence Detector on the FPGA Kit .