22bec121

Experiment – 9 Date:-21/3/2024

Lab Work

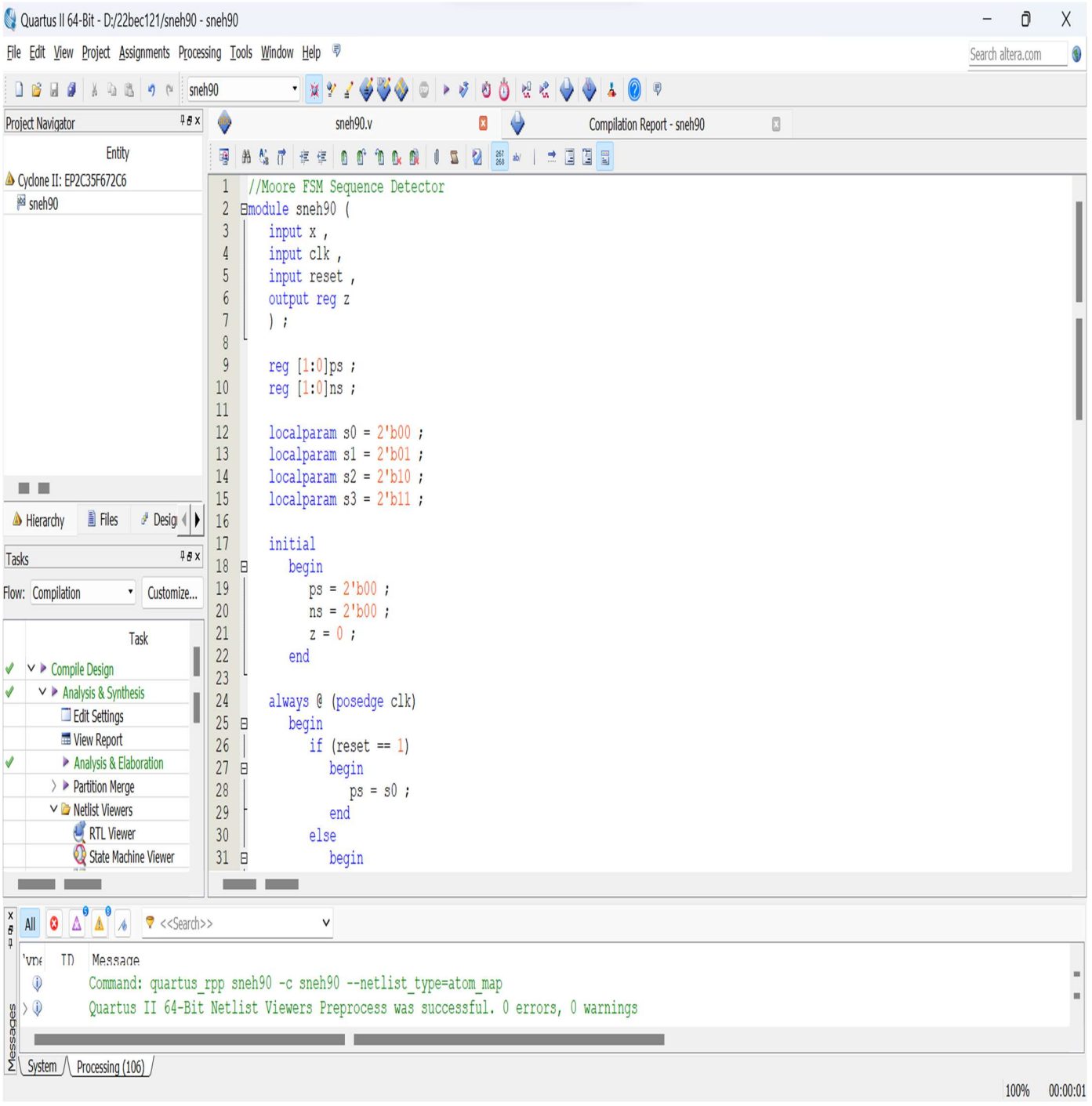
Q1 .Design a Moore Based FSM 101 Overlapping Sequence Detector .

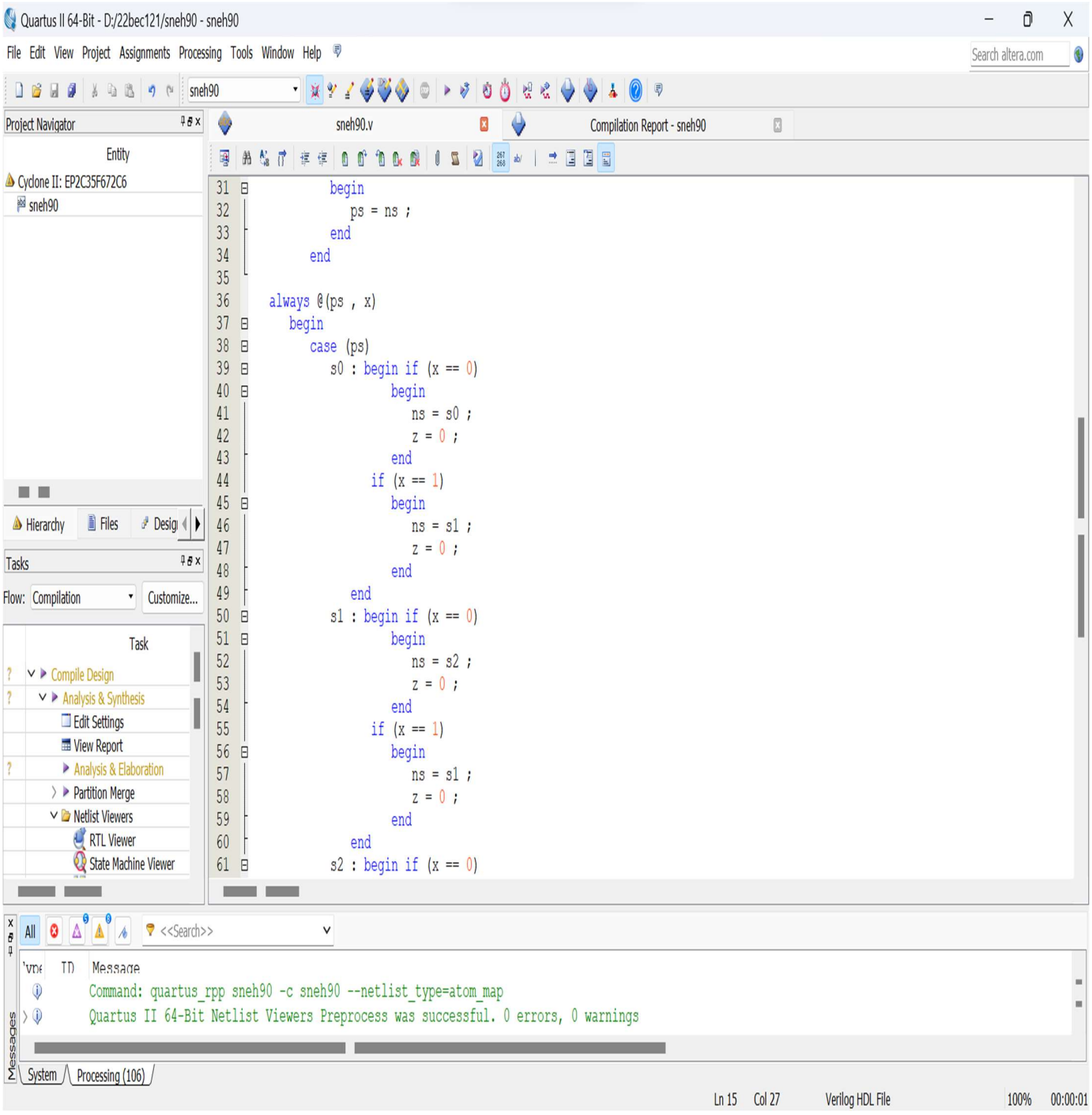
Code (Moore FSM)

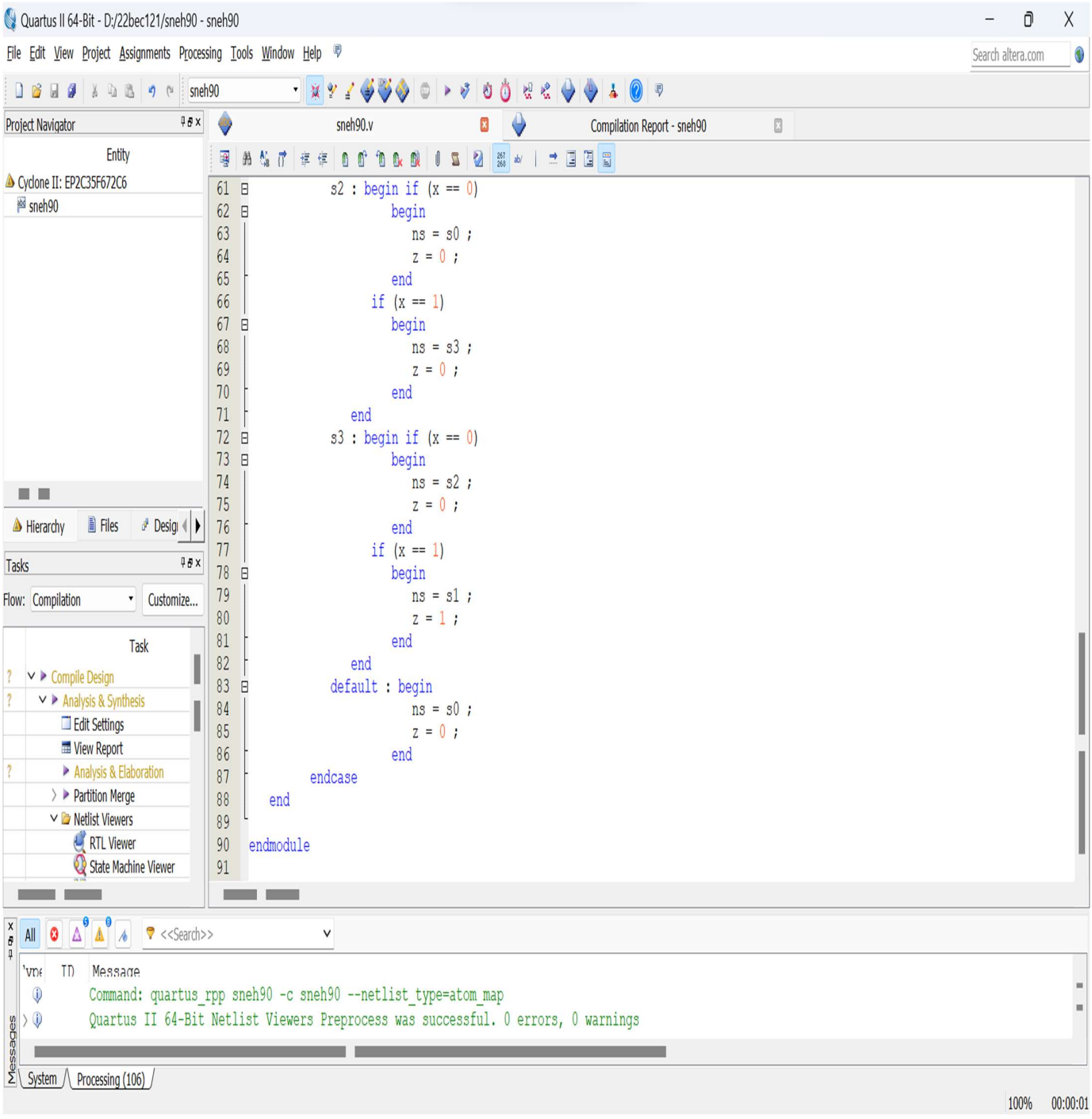
```
//Moore FSM Sequence Detector
module sneh90 (
        input x ,
        input clk ,
        input reset ,
        output reg z
        ) ;

        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 ;
```

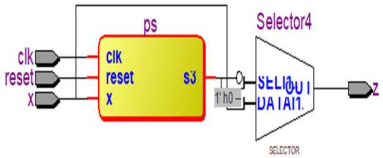
```
                                        z = 0 ;
                                    end
                                end
                        s2 : begin if (x == 0)
                                    begin
                                        ns = s0 ;
                                        z = 0 ;
                                    end
                                if (x == 1)
                                    begin
                                        ns = s3 ;
                                        z = 0 ;
                                    end
                                end
                        s3 : begin if (x == 0)
                                    begin
                                        ns = s2 ;
                                        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/sneh90 - sneh90
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh90
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh90
sneh90.v
Compilation Report - sneh90
1 //Moore FSM Sequence Detector
2 module sneh90 (
3 input x ,
4 input clk ,
5 input reset ,
6 output reg z
7) ;
8
9 reg [1:0]ps ;
10 reg [1:0]ns ;
11
12 localparam s0 = 2'b00 ;
13 localparam s1 = 2'b01 ;
14 localparam s2 = 2'b10 ;
15 localparam s3 = 2'b11 ;
16
17 initial
18 begin
19 ps = 2'b00 ;
20 ns = 2'b00 ;
21 z = 0 ;
22 end
23
24 always @ (posedge clk)
25 begin
26 if (reset == 1)
27 begin
28 ps = s0 ;
29 end
30 else
31 begin
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
Command: quartus_rpp sneh90 -c sneh90 --netlist_type=atom_map
Quartus II 64-Bit Netlist Viewers Preprocess was successful. 0 errors, 0 warnings
Messages
System
Processing (106)
100%
00:00:01

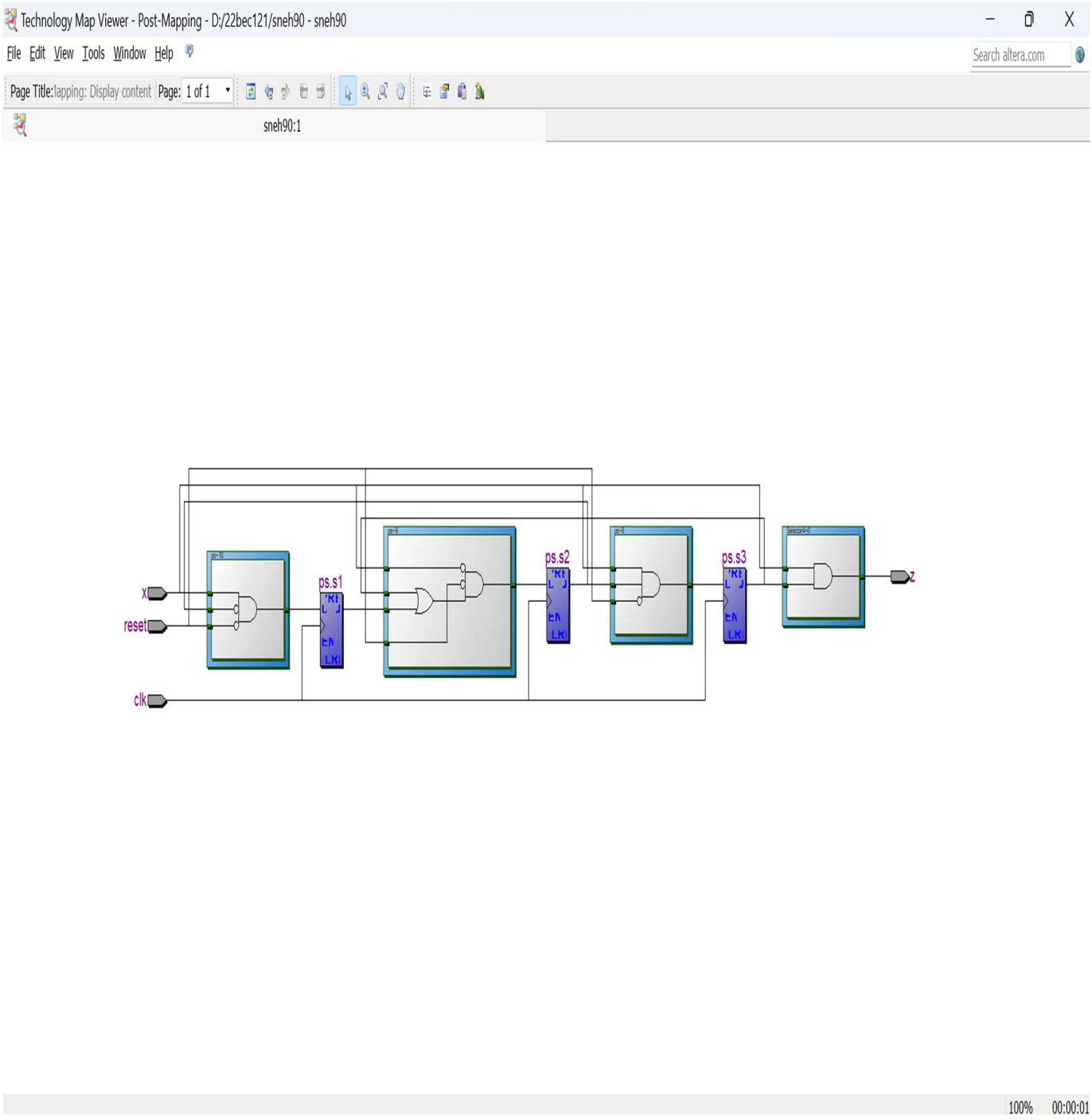

Quartus II 64-Bit - D:/22bec121/sneh90 - sneh90
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh90
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh90
sneh90.v
Compilation Report - sneh90
begin
ps = ns ;
end
end
always @(ps , x)
begin
case (ps)
s0 : begin if (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)
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>>
Message
Command: quartus_rpp sneh90 -c sneh90 --netlist_type=atom_map
Quartus II 64-Bit Netlist Viewers Preprocess was successful. 0 errors, 0 warnings
Messages
System
Processing (106)
Ln 15 Col 27 Verilog HDL File 100% 00:00:01


Quartus II 64-Bit - D:/22bec121/sneh90 - sneh90
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
sneh90
Project Navigator
Entity
Cyclone II: EP2C35F672C6
sneh90
sneh90.v
Compilation Report - sneh90
61 s2 : begin if (x == 0)
62 begin
63 ns = s0 ;
64 z = 0 ;
65 end
66 if (x == 1)
67 begin
68 ns = s3 ;
69 z = 0 ;
70 end
71 end
72 s3 : begin if (x == 0)
73 begin
74 ns = s2 ;
75 z = 0 ;
76 end
77 if (x == 1)
78 begin
79 ns = s1 ;
80 z = 1 ;
81 end
82 end
83 default : begin
84 ns = s0 ;
85 z = 0 ;
86 end
87 endcase
88 end
89
90 endmodule
91
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
Command: quartus_rpp sneh90 -c sneh90 --netlist_type=atom_map
Quartus II 64-Bit Netlist Viewers Preprocess was successful. 0 errors, 0 warnings
Messages
System
Processing (106)
100%
00:00:01

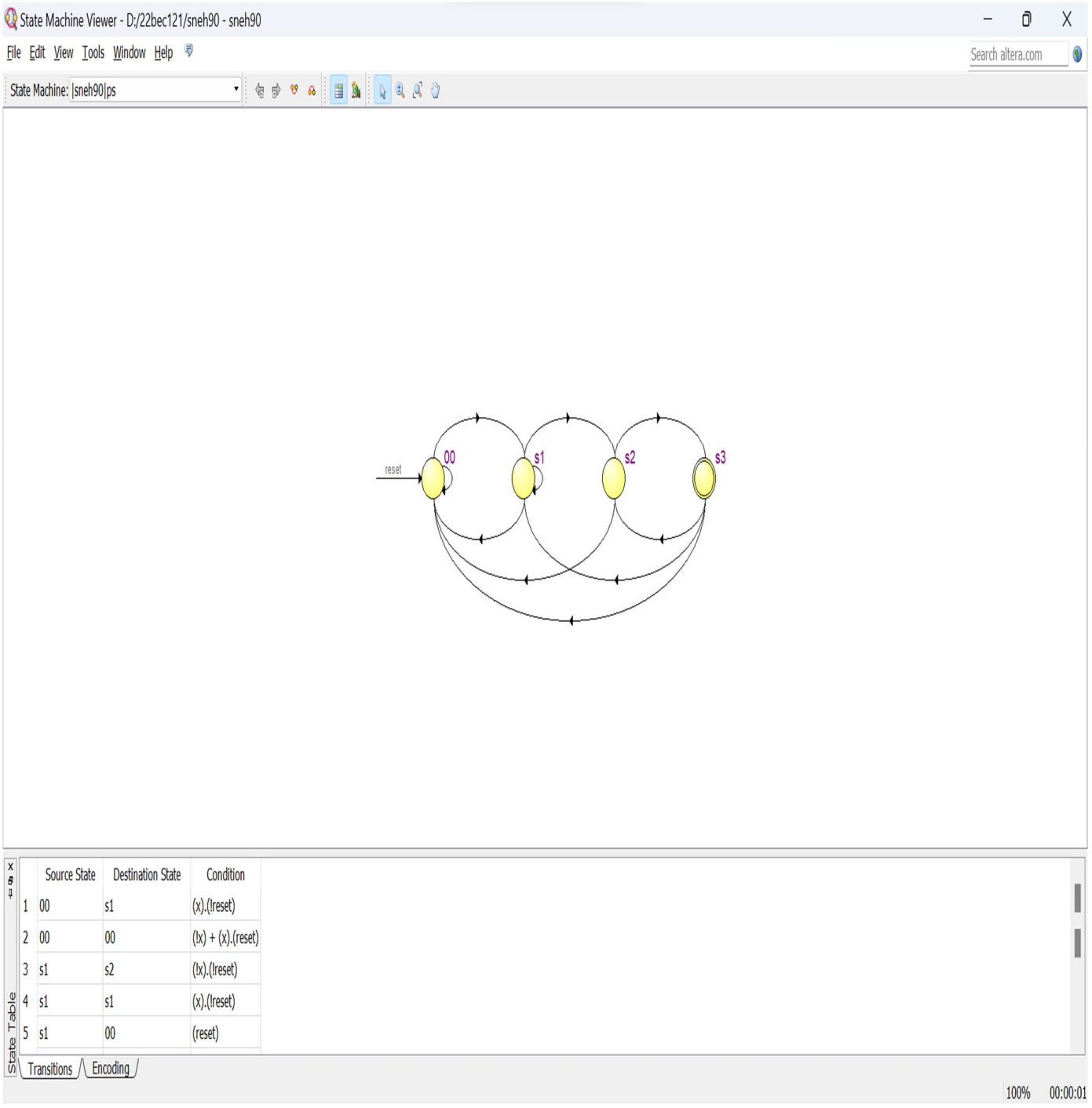
Output 1)Cyclone II (RTL View)



2)Cyclone II (TTL View)

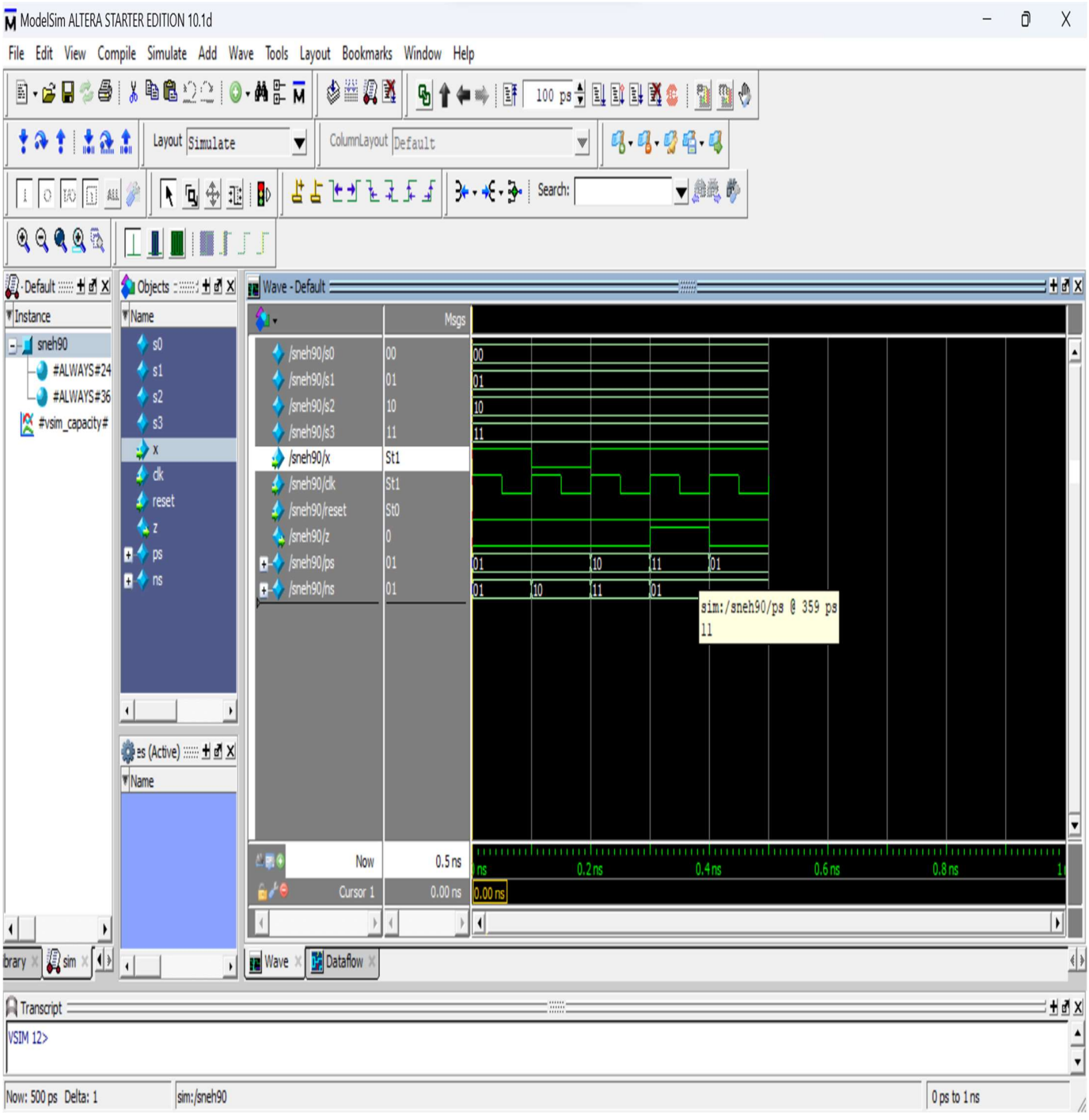


3) State Diagram

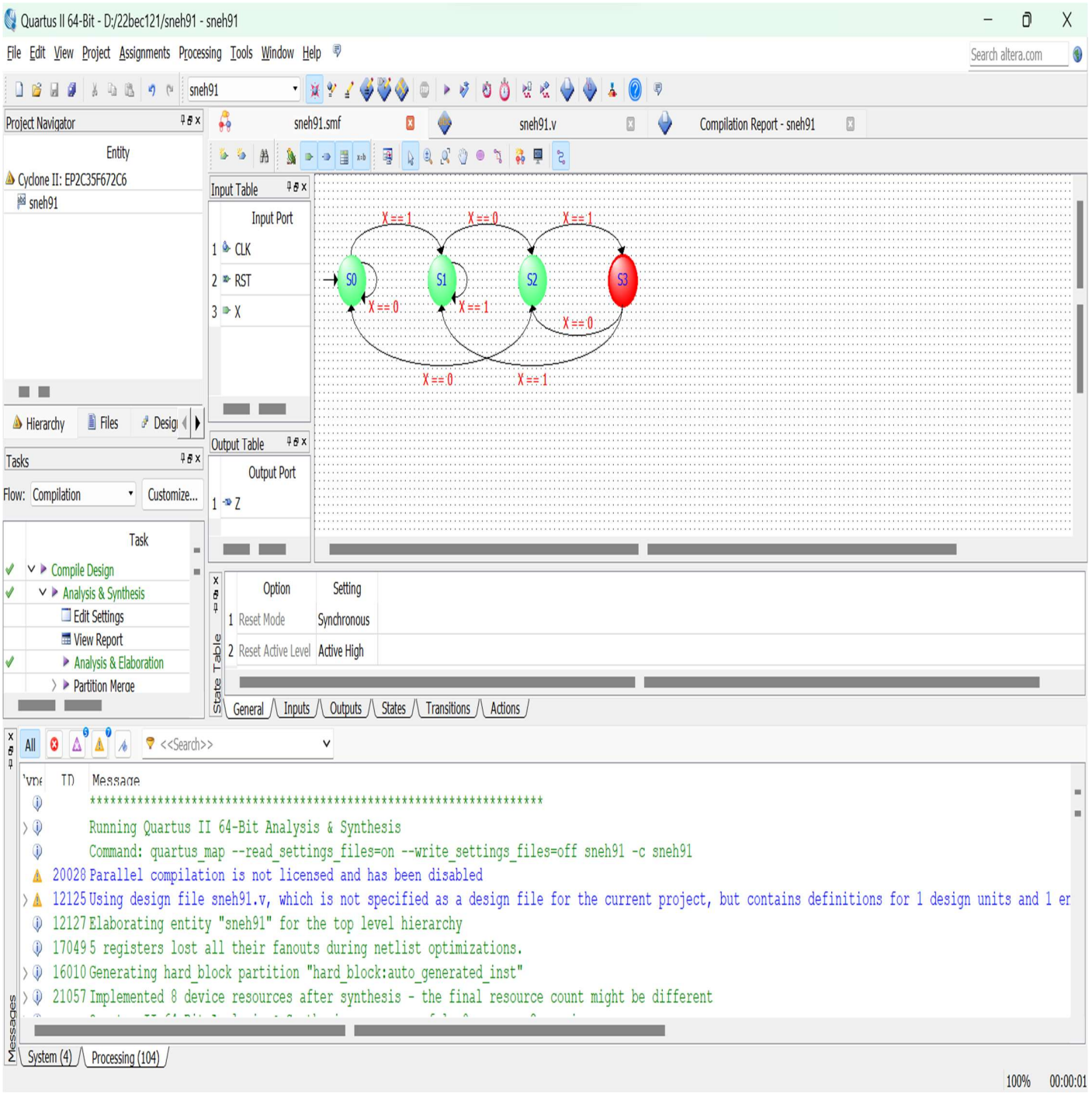


	Source State	Destination State	Condition
1	00	s1	(x).(!reset)
2	00	00	(!x) + (x).(reset)
3	s1	s2	(!x).(!reset)
4	s1	s1	(x).(!reset)
5	s1	00	(reset)

Simulation Results :-



State Machine Wizard



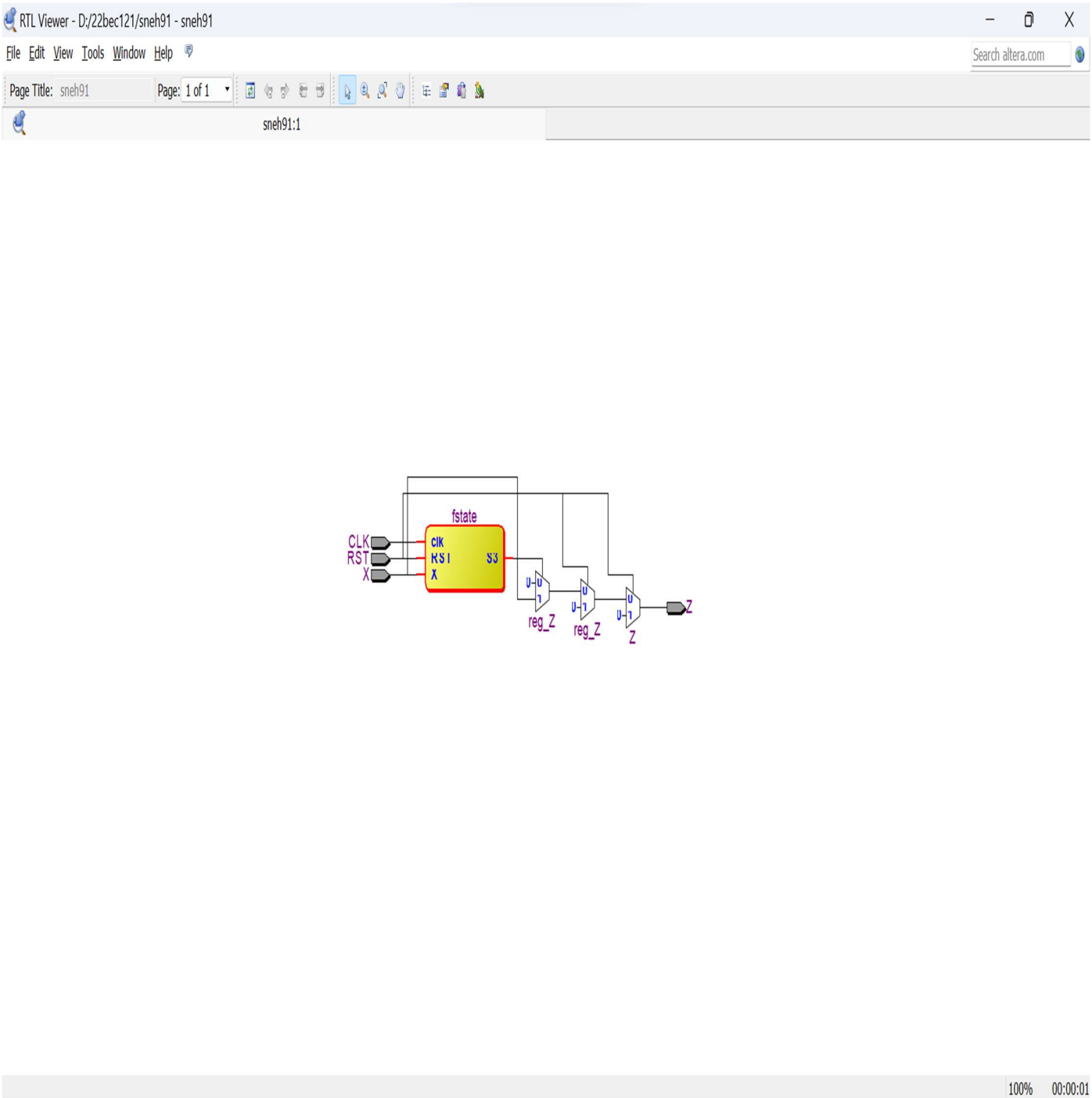
Generated Code

```
// 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 28 07:09:48 2024

// synthesis message_off 10175

`timescale 1ns/1ns

module sneh91 (
    CLK,RST,X,
    Z);

    input CLK;
    input RST;
    input X;
    tri0 RST;
    tri0 X;
    output Z;
    reg Z;
    reg reg_Z;
    reg [3:0] fstate;
    reg [3:0] reg_fstate;
    parameter S0=0,S1=1,S2=2,S3=3;
```

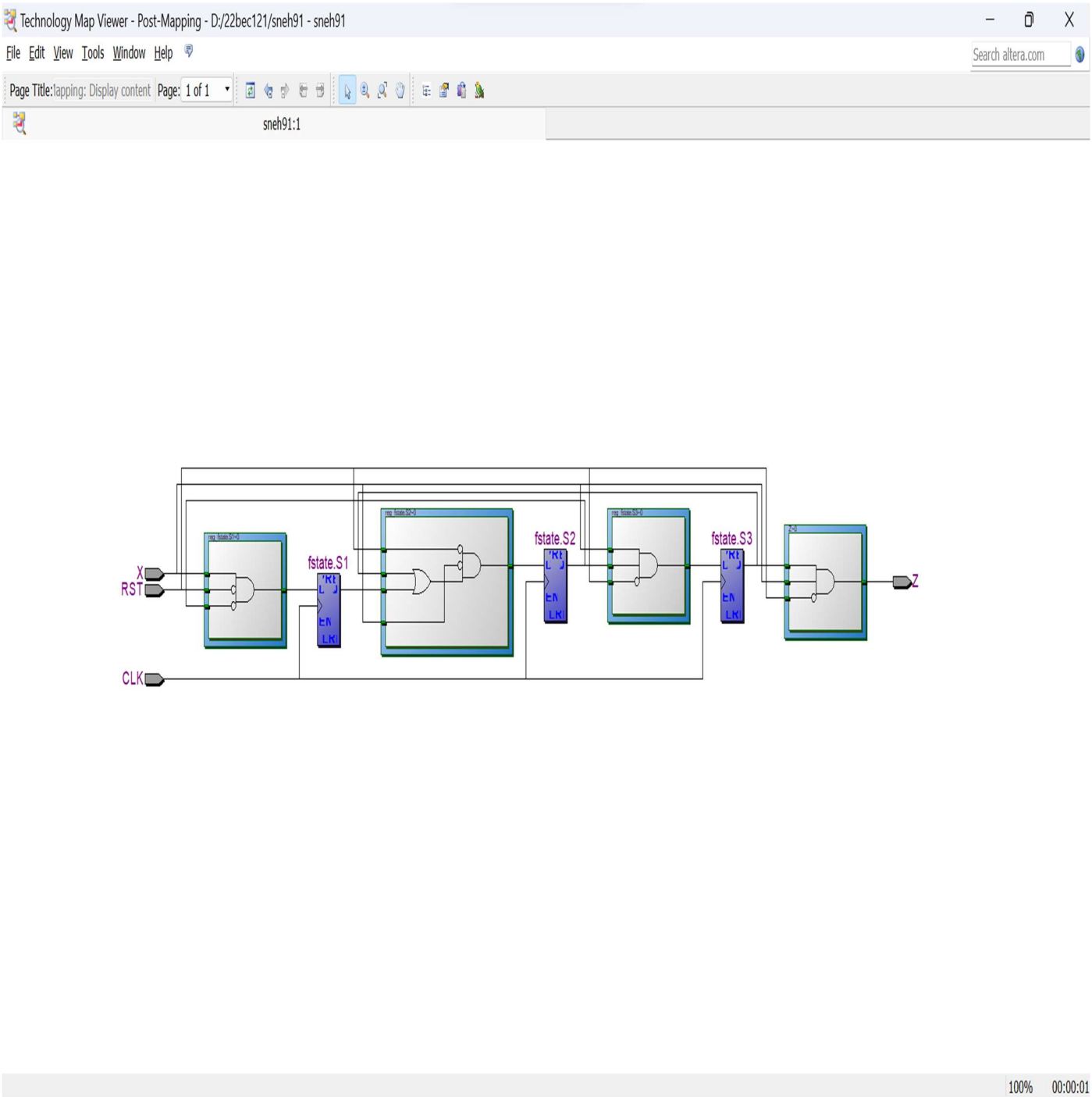
```
initial
begin
   reg_Z <= 1'b0;
end

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

always @(fstate or RST or X or reg_Z)
begin
   if (RST) begin
      reg_fstate <= S0;
      reg_Z <= 1'b0;
      Z <= 1'b0;
   end
   else begin
      reg_Z <= 1'b0;
      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;
         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;
            end
            S2: begin
               if ((X == 1'b0))
                  reg_fstate <= S0;
               else if ((X == 1'b1))
                  reg_fstate <= S3;
               // Inserting 'else' block to prevent latch inference
               else
                  reg_fstate <= S2;
            end
            S3: begin
               if ((X == 1'b1))
                  reg_fstate <= S1;
               else if ((X == 1'b0))
                  reg_fstate <= S2;
               // Inserting 'else' block to prevent latch inference
               else
                  reg_fstate <= S3;

               if ((X == 1'b1))
                  reg_Z <= 1'b1;
               // Inserting 'else' block to prevent latch inference
               else
                  reg_Z <= 1'b0;
            end
            default: begin
               reg_Z <= 1'bx;
               $display ("Reach undefined state");
            end
         endcase
         Z <= reg_Z;
      end
   end
endmodule // sneh91
```

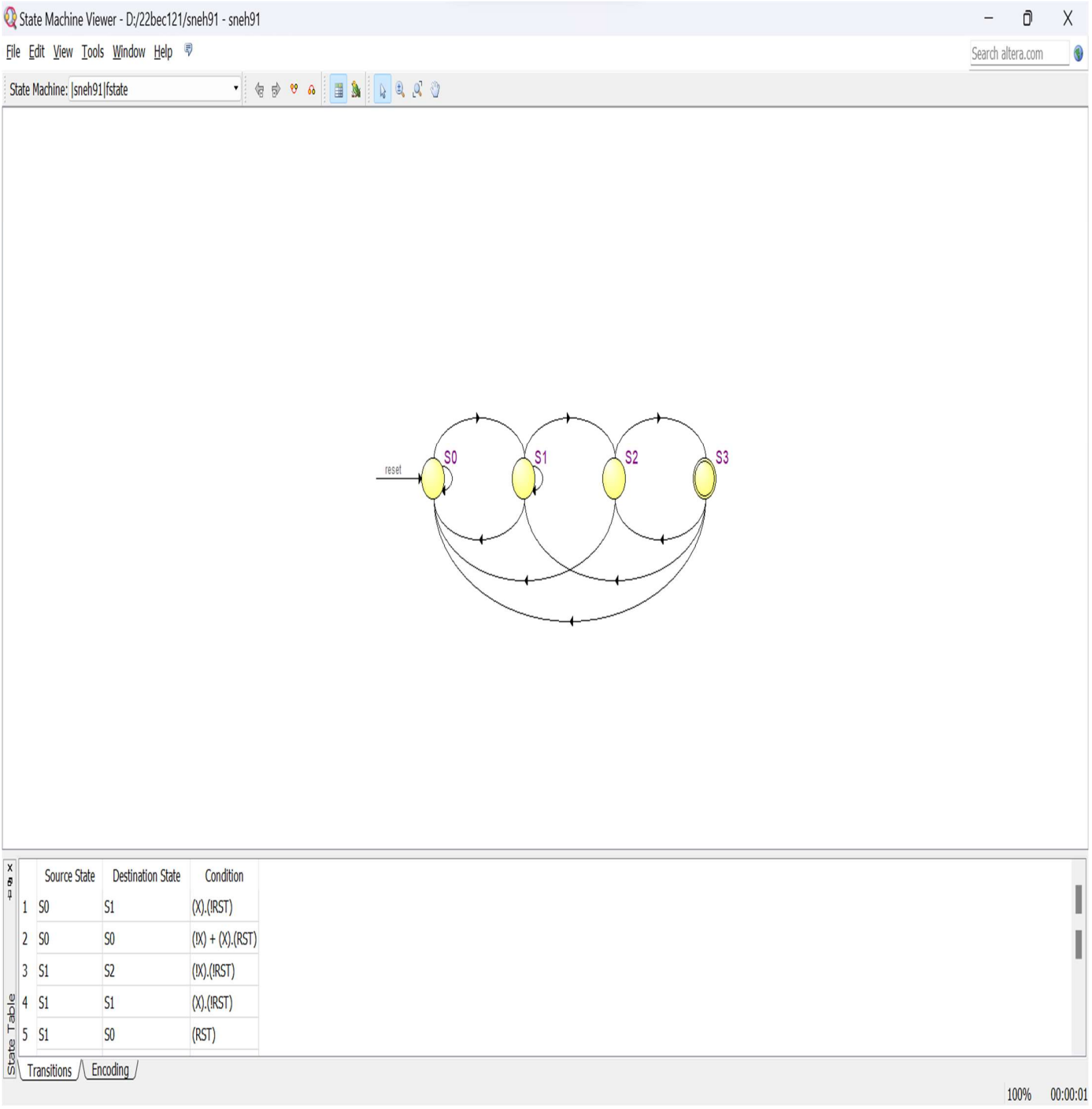
Output 1)Cyclone II (RTL View)



2)Cyclone II (TTL View)



3)State Machine View



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

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

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

We also learnt how RTL and TTL forms for a Moore 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 Moore FSM Sequence Detector on the FPGA Kit .