

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
    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
        end
    end

```

```

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

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 ;
```

Tasks

Flow: Compilation Customize...

Task

- ✓ Compile Design
- ✓ Analysis & Synthesis
 - Edit Settings
 - View Report
- ✓ Analysis & Elaboration
- Partition Merge
- Netlist Viewers
 - RTL Viewer
 - State Machine Viewer

Messages

All

Message

Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings

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

Hierarchy Files Design

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*

```
29 else
30     begin
31         ps = ns ;
32     end
33 end
34
35 always @(ps , x)
36 begin
37     case (ps)
38     s0 : begin if (x == 0)
39         begin
40             ns = s0 ;
41             z = 0 ;
42         end
43         if (x == 1)
44         begin
45             ns = s1 ;
46             z = 0 ;
47         end
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 end
```

Compilation Report - sneh78

Messages

All

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

Hierarchy Files Design

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*

```
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
```

Compilation Report - sneh78

Messages

All

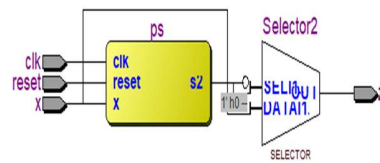
Message

Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings

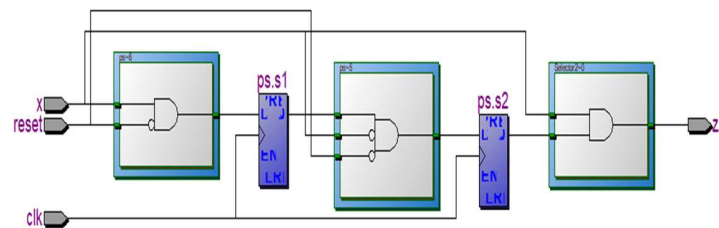
System (2) / Processing (98)

Ln 68 Col 32 Verilog HDL File 100% 00:00:16

Output 1)Cyclone II (RTL View)



2)Cyclone II (TTL View)



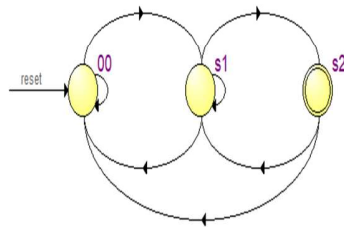
3) State Diagram

State Machine Viewer - D:/22bec121/sneh78 - sneh78

File Edit View Tools Window Help

Search altera.com

State Machine: |sneh78|.ps

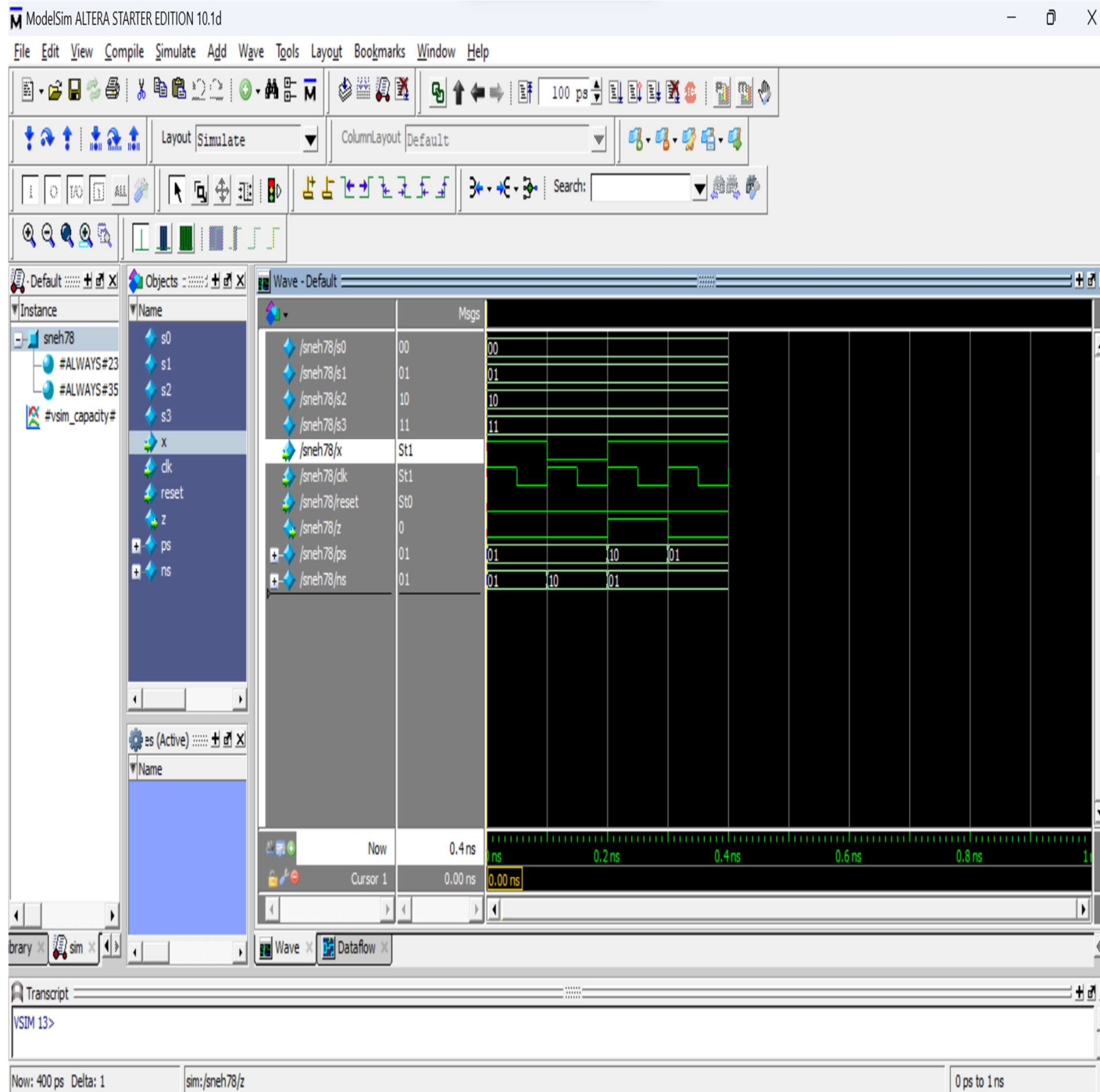


x s 1	State Table		
	Source State	Destination State	Condition
	1 00	s1	(x).(reset)
	2 00	00	(x) + (x).(reset)
	3 s1	s2	(x).(reset)
	4 s1	s1	(x).(reset)
	5 s1	00	(reset)

Transitions / Encoding /

0% 00:00:00

Simulation Results :-



State Machine Wizard

Quartus II 64-Bit - C:/WORK/22bec121/sneh89 - sneh89

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

sneh89

Project Navigator

Cyclone II: EP2C35F67

sneh89

Input ...

Input Port

1 CLK

2 RST

3 X

Output ...

Output Port

1 Z

State Machine Diagram

States: S0, S1, S2

Transitions:

- S0 to S0: X==0
- S0 to S1: X==1
- S1 to S1: X==1
- S1 to S2: X==0
- S2 to S1: X==1
- S2 to S0: X==0

Transition (In Verilog or VHDL 'OTHERS')

Source State	Destination State	Transition
S0	S0	X==0
S1	S1	X==1

General Inputs Outputs States Transitions Actions

Messages

Quartus II 64-Bit Netlist Viewers Preprocess was successful. 0 errors, 0 warnings

System (10) / Processing (101)

100% 00:00

NIFTY +0.60%

Search

ENG IN

12:35:18 21-03-2024

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))
```

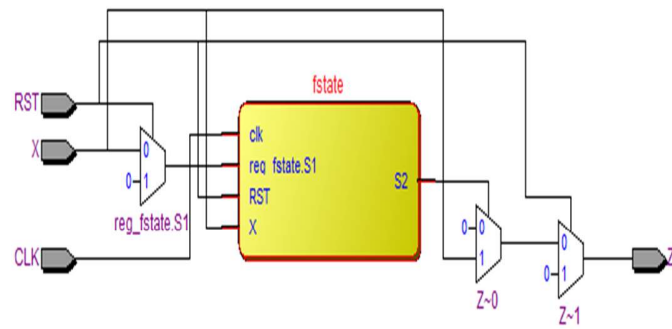
```

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

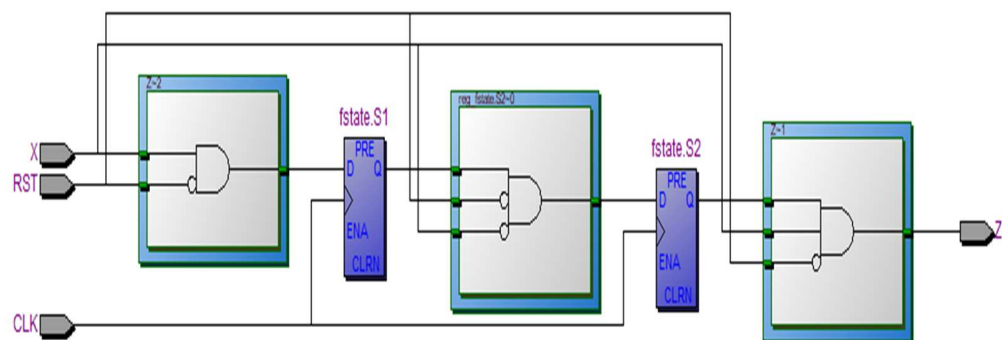
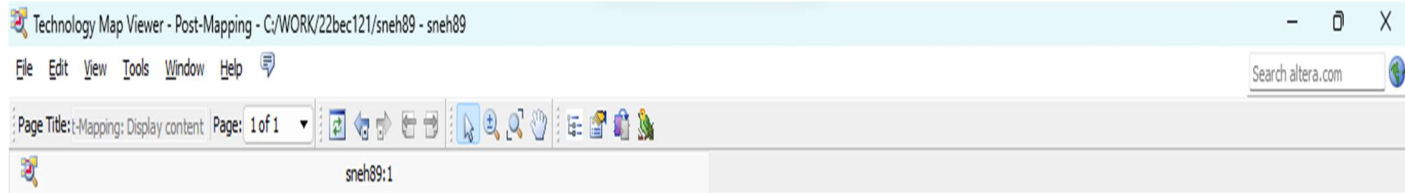
    if ((X == 1'b1))
        Z <= 1'b1;
    // Inserting 'else' block to prevent latch inference
    else
        Z <= 1'b0;
    end
    default: begin
        Z <= 1'bx;
        $display ("Reach undefined state");
    end
endcase
end
end
endmodule // sneh89

```

The image shows the top portion of the RTL Viewer application. The title bar indicates the file path 'C:/WORK/22bec121/sneh89 - sneh89'. The menu bar includes 'File', 'Edit', 'View', 'Tools', 'Window', and 'Help'. The toolbar contains various icons for navigation and editing. The status bar at the bottom shows 'Page Title: sneh89', 'Page: 1 of 1', and 'sneh89:1'.



2)Cyclone II (TTL View)



3)State Machine View

State Machine Viewer - C:/WORK/22bec121/sneh89 - sneh89

File Edit View Tools Window Help

State Machine: |sneh89|fstate

```
graph LR; reset(( )) -- reset --> S0((S0)); S0 -- "(!X) + (X). (RST)" --> S0; S0 -- "(!X). (!RST)" --> S1((S1)); S1 -- "(!X). (!RST)" --> S2(((S2))); S2 -- "(!X) + (X). (RST)" --> S0;
```

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)

Transitions / Encoding

0% 00:00:00

34°C Smoke

Q Search

ENG IN

12:36:19 21-03-2024

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 .