Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2021-2022

ECE5017 - Digital Design with FPGA ELA

M.Tech VLSI Design

School of Electronics Engineering

Vellore Institute of Technology

Name: Shreyas S Bagi

Register Number: 21MVD0086

Slot: L5+L6

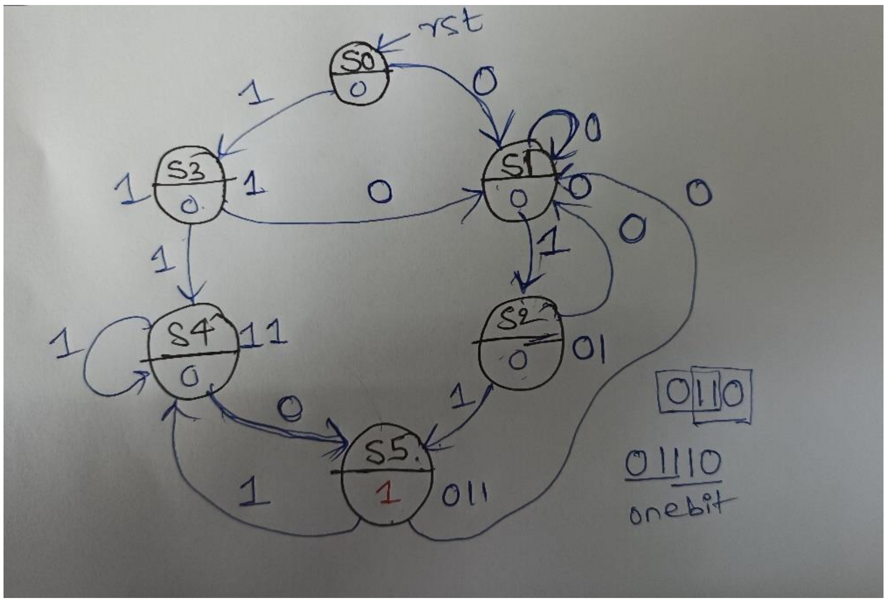
Date:31/12/2021

Lab Task 04

Finite State Machine Modelling

Aim: Construct a sequence detector that accepts a binary number entered one bit at a time, most significant bit first, and indicates the output with a LED light if the sequence are of {011,110}. Implement this design on Altera DE2-115 FPGA Kit using Quartus II.

State Diagram :



Objective:

- To design the sequence detector using Finite State Machine.
- To write the test bench for generating stimulus.
- To perform functional simulation.
- Understand realization of the FSM in FPGA boards.
- The FSM is implemented using Moore logic and One-hot encoding for states.

Software and Hardware Details:

For design Functional Simulation: Modelsim-INTEL FPGA STARTER EDITION.

For Implementation of the Function using ALTERA Quartus Prime and DE2-115 FPGA Kit (Cyclone IV E: EP4CE115F29C7).

Verilog Code :

module Sequence_detect (clk, clk_out, rst, din, out);

input clk, rst, din;

Date:31/12/2021

```
output clk_out;

output reg out;


/*   The Meaning of the Input ports and output is explained in this comments what is
the purpose of it.

    1. clk = CLK 50 MHZ internal Clock PIN_Y2 is the PIN Location of FPGA.
    2. din is the Switch button corresponding Pin location is PIN_AC28 (SW1 in
       Board).
    3. rst is the Switch button corresponding Pin location is PIN_AB28 (SW0 in
       Board).
    4. clk_out is the Red LED which turns on for every one second. Clk_out is
       assigned to PIN_G19
    5. out is the Green LED which turns on when 011 and 110 sequence is detected.
       The overlapping sequence is also detected.

*/

parameter S0=6'b000001, S1=6'b000010,S2=6'b000100,S3=6'b001000,
S4=6'b010000, S5=6'b100000;

/*

Moore Logic is better than Mealy Logic because output have full cycle to settle
through the combinational logic and Moore state machine are favoured in Industry.
The output is only function of Present state (PS). The State are encoded using One-
hot encoding here 6 states so 6 Flip flop are realized

*/

reg [5:0] PS,NS;

// 1 Hz Clock divider circuit instantiation

Clock_Division GA1 (clk,rst,clk_out);

// Change the State only at Positive edge of the 1 second clock.

always @(posedge clk_out)

begin

if(rst)

PS <= S0;

else

PS <= NS;

end

// Change the Next State for changes w.r.t to Present state and Data input.
```

Date:31/12/2021

```
always @(PS,din)

begin

case(PS)

S0: NS <= din?S3:S1;

S1: NS <= din?S2:S1;

S2: NS <= din?S5:S1;

S3: NS <= din?S4:S1;

S4: NS <= din?S4:S5;

S5: NS <= din?S4:S1;

default : NS <= S0;

endcase

end

/*

Change the Output if changes in the Present state because in Moore logic output is
only function of Present state.

*/

always @(PS)

begin

case(PS)

S0: out = 1'b0;

S1: out = 1'b0;

S2: out = 1'b0;

S3: out = 1'b0;

S4: out = 1'b0;

S5: out = 1'b1;

default : out = 1'b0;

endcase

end

endmodule

// The Module used for Clock Divider circuit.

module Clock_Division(CLK, reset, LED);
```

Date:31/12/2021

```
input CLK, reset;
output reg LED = 1'b0;
reg [25:0] CLK_DIV;
always @(posedge CLK)
begin
if(CLK_DIV == 26'b1011_1110_1011_1100_0010_0000_00)
begin
if (reset)
LED <= 0;
else
LED <= ~LED;
CLK_DIV <= 26'b0;
end
else
CLK_DIV = CLK_DIV + 1;
end
endmodule
```

Testbench Code :

```
module Sequence_task1_tb();
reg clk,rst,din;
wire out;
Sequence_task1 GA1 (clk,rst,din,out);
initial
begin
clk=1'b0;
rst = 1;
#1 rst = 0;
$monitor($time,"Clk=%b,rst=%b,din=%b,out=%b,PS=%b,NS=%b", clk, rst, din,
out, Sequence_task1_tb.GA1.PS, Sequence_task1_tb.GA1.NS);
end
```

Date:31/12/2021

```
always #5 clk = (~clk);
initial
begin
//#12 din = 0;
//
#12  din = 1;
#10  din = 0;
#10  din = 1;
#10  din = 1;
#10  din = 0;
#10  din = 1;
#10  din = 1;
#10  din = 0;
#5 $stop;
end
endmodule
```

Simulation Waveform :

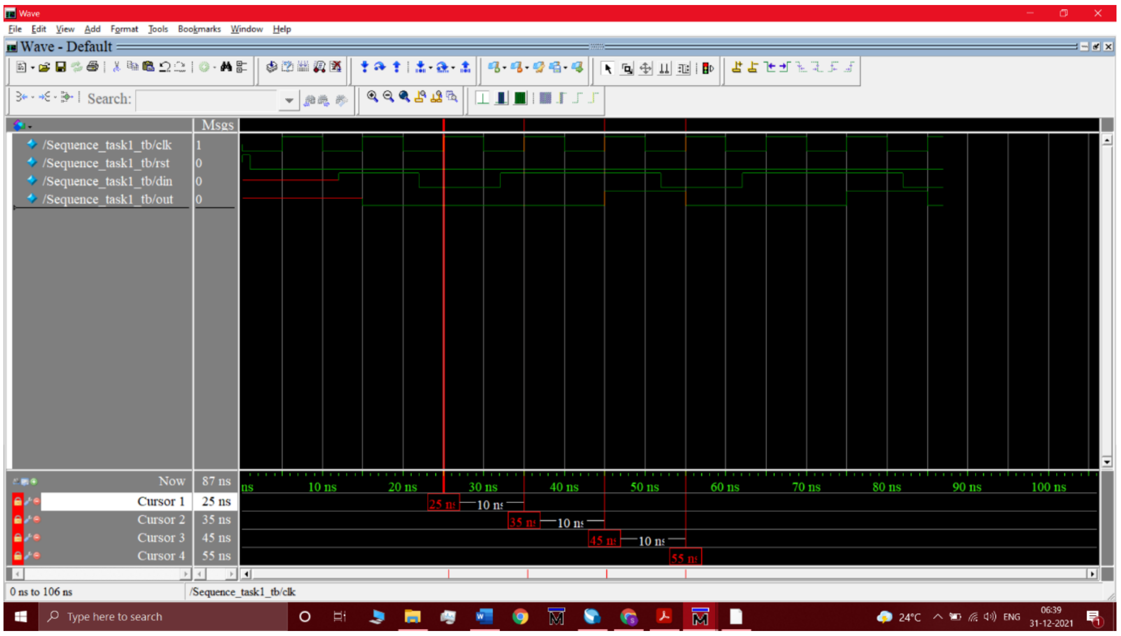
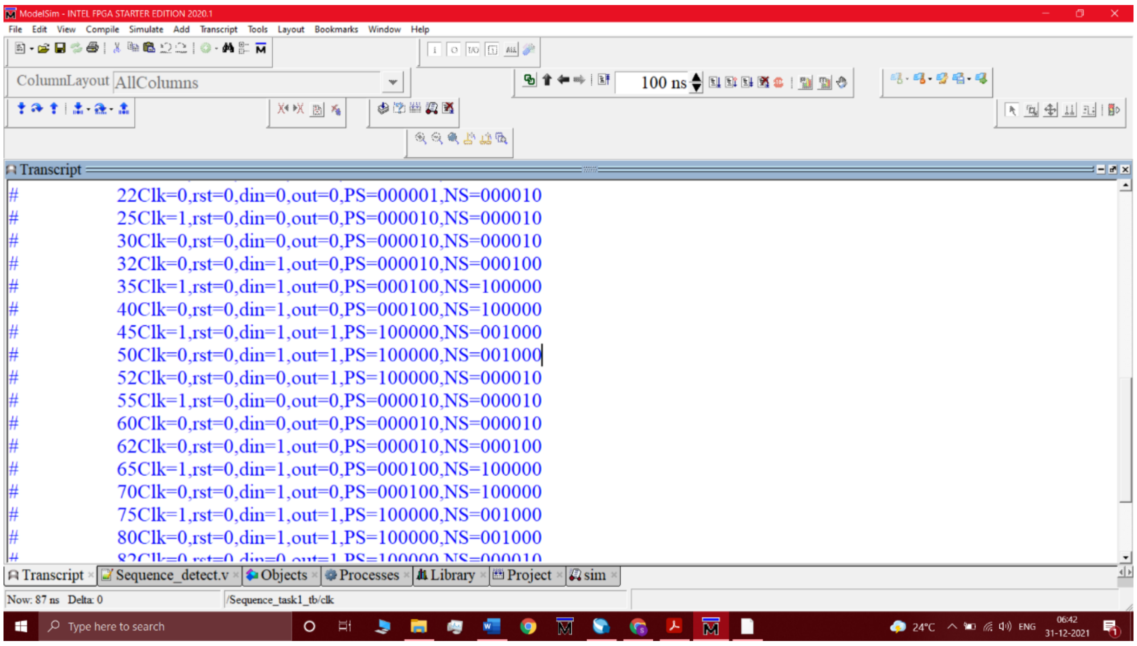


Figure 1.1 Simulation result for Sequence Detector at every positive edge of the clock the input is read.

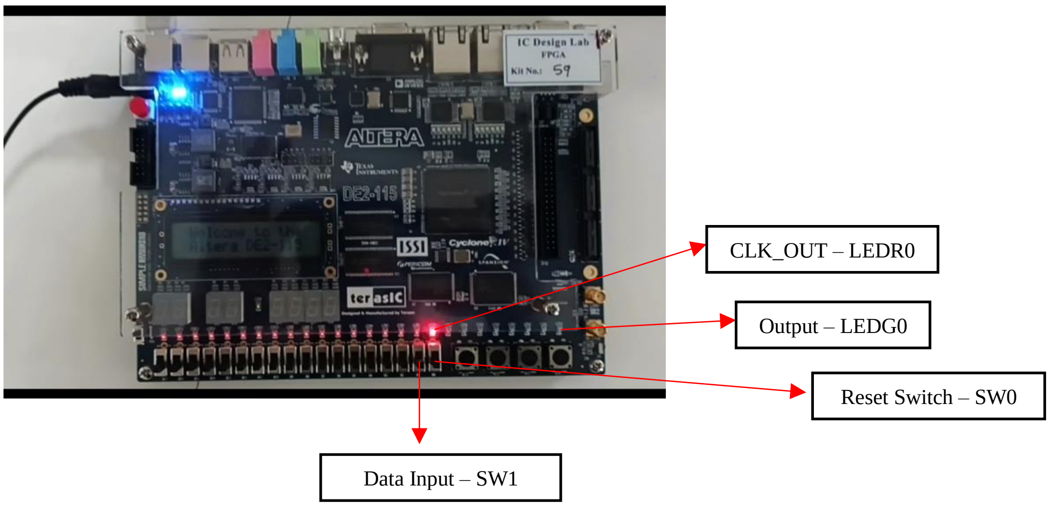
Date:31/12/2021

In the Figure 1.1 we can observe at 25ns the data input is read as 0, next at 35 ns the data input read as 1 at 45ns the data input read as 1. So, the output becomes 1 upto next positive logic. Therefore, from 45ns to 55ns the output remains HIGH and at 55ns it detected data input as 1 so, the sequence is now 111 so it goes low.


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
ColumnLayout AllColumns
100 ns
Transcript
Transcript Sequence_detect.v Objects Processes Library Project sim
Now: 87 ns Delta: 0
/Sequence_task1_tb/clk

```
#            22Clk=0,rst=0,din=0,out=0,PS=000001,NS=000010
#            25Clk=1,rst=0,din=0,out=0,PS=000010,NS=000010
#            30Clk=0,rst=0,din=0,out=0,PS=000010,NS=000010
#            32Clk=0,rst=0,din=1,out=0,PS=000010,NS=000100
#            35Clk=1,rst=0,din=1,out=0,PS=000100,NS=100000
#            40Clk=0,rst=0,din=1,out=0,PS=000100,NS=100000
#            45Clk=1,rst=0,din=1,out=1,PS=100000,NS=001000
#            50Clk=0,rst=0,din=1,out=1,PS=100000,NS=001000
#            52Clk=0,rst=0,din=0,out=1,PS=100000,NS=000010
#            55Clk=1,rst=0,din=0,out=0,PS=000010,NS=000010
#            60Clk=0,rst=0,din=0,out=0,PS=000010,NS=000010
#            62Clk=0,rst=0,din=1,out=0,PS=000010,NS=000100
#            65Clk=1,rst=0,din=1,out=0,PS=000100,NS=100000
#            70Clk=0,rst=0,din=1,out=0,PS=000100,NS=100000
#            75Clk=1,rst=0,din=1,out=1,PS=100000,NS=001000
#            80Clk=0,rst=0,din=1,out=1,PS=100000,NS=001000
#            82Clk=0,rst=0,din=0,out=1,PS=100000,NS=000010
```

Figure 1.2 Simulation result for Sequence Detector at every positive edge of the clock the data input is read and corresponding Present state and Next state how the state transition happens is displayed using $monitor command.



Date:31/12/2021

Figure 1.3 The Code dumped or programmed to DE2-115 Board and The LEDR0 is continuously toggling for 1 second**.**

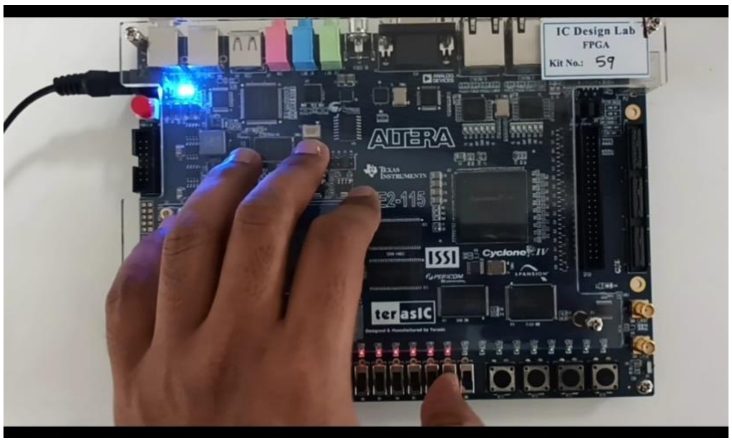


Figure 1.4 The user is providing the Data input for sequence. If SW1 is pushed up read as Input High and pulled low read as High. At every positive clock cycle the data input is read.

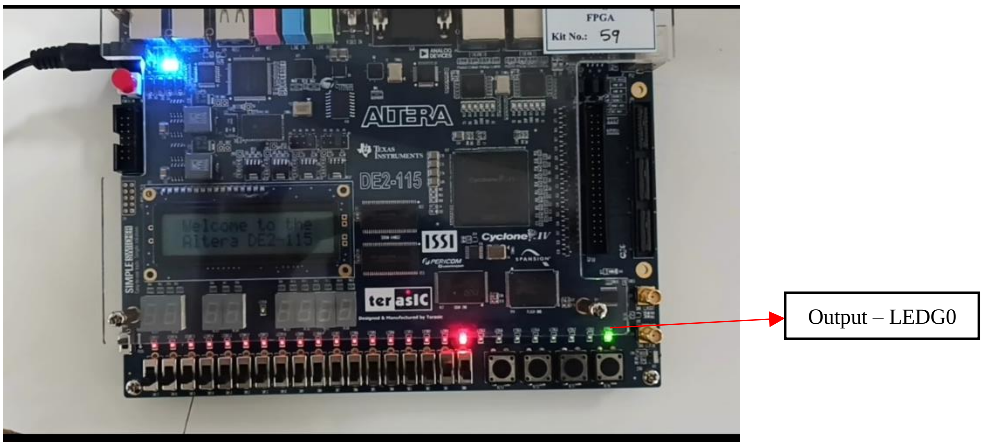


Figure 1.4 The data input is One continuously actually stating read 0 then user changed to 1 and then for two positive edge clk it read as one. Therefore, sequence detected. The LEDG0 is made high.

Date:31/12/2021

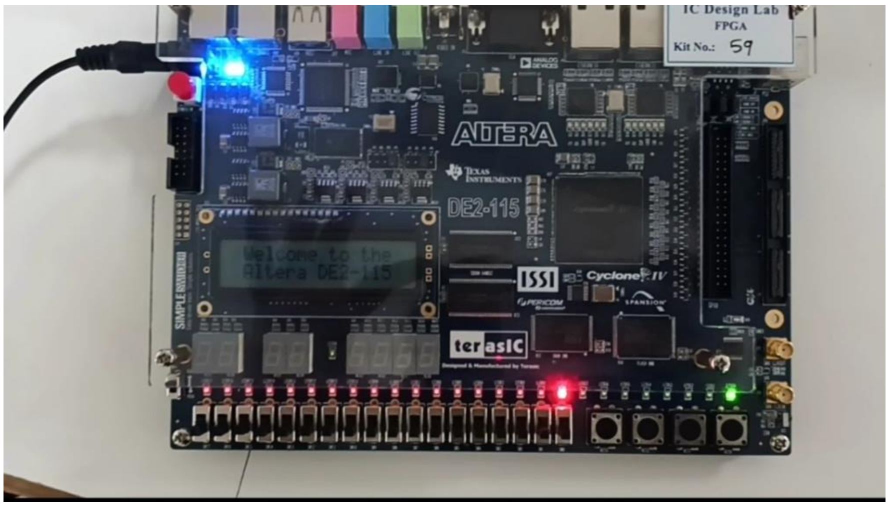
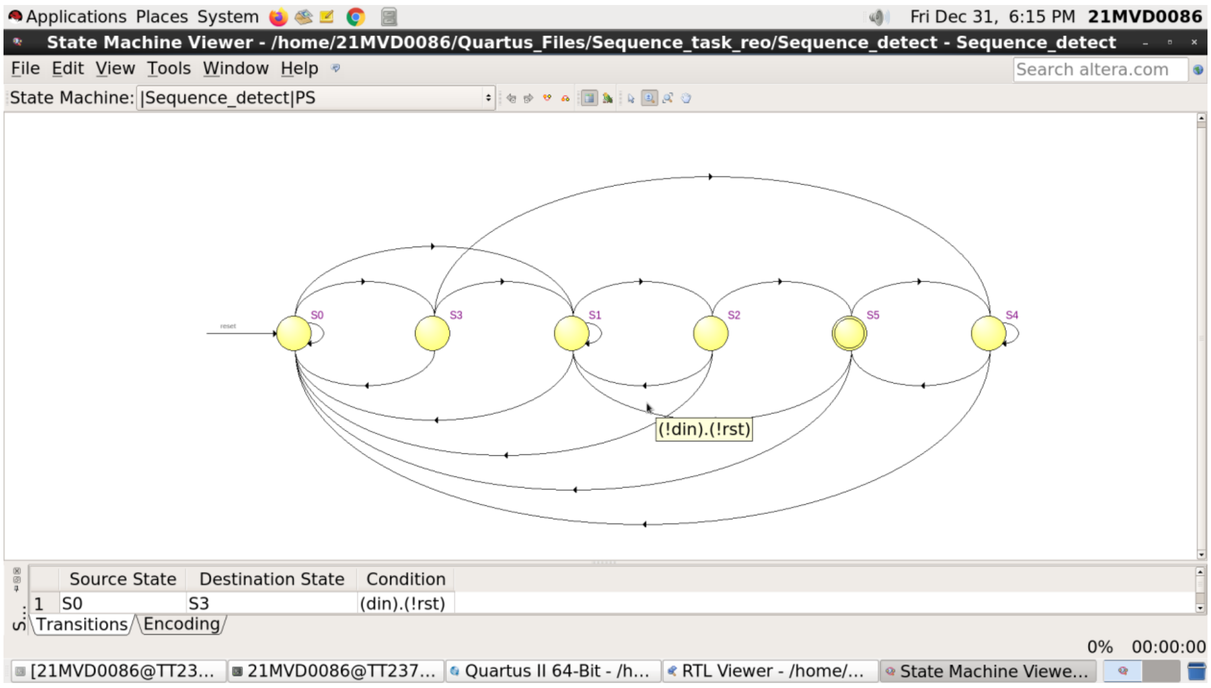


Figure 1.5 The data input is Zero. So, the sequence from previous was 11 and 0 now. Therefore, the sequence is detected and LED Green is set to HIGH.

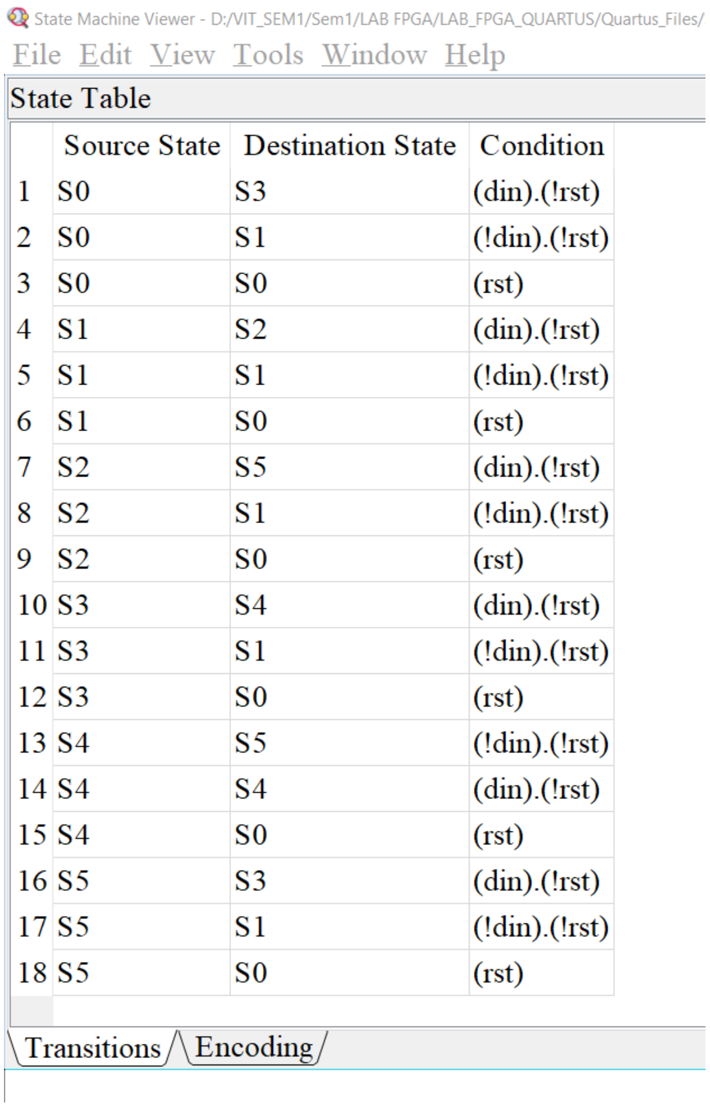
State Machine realized in Quartus Prime:


Applications Places System
Fri Dec 31, 6:15 PM 21MVD0086
State Machine Viewer - /home/21MVD0086/Quartus_Files/Sequence_task_reo/Sequence_detect - Sequence_detect
File Edit View Tools Window Help
Search altera.com
State Machine: |Sequence_detect|PS
reset
S0
S3
S1
S2
S5
S4
(!din).(!rst)
Transitions
Encoding
0% 00:00:00
[21MVD0086@TT23...
21MVD0086@TT237...
Quartus II 64-Bit - /h...
RTL Viewer - /home/...
State Machine Viewe...

	Source State	Destination State	Condition
1	S0	S3	(din).(!rst)

Date:31/12/2021

State Table realized in Quartus Prime:


State Machine Viewer - D:/VIT_SEM1/Sem1/LAB FPGA/LAB_FPGA_QUARTUS/Quartus_Files/
File Edit View Tools Window Help
State Table
Transitions
Encoding

	Source State	Destination State	Condition
1	S0	S3	(din).(!rst)
2	S0	S1	(!din).(!rst)
3	S0	S0	(rst)
4	S1	S2	(din).(!rst)
5	S1	S1	(!din).(!rst)
6	S1	S0	(rst)
7	S2	S5	(din).(!rst)
8	S2	S1	(!din).(!rst)
9	S2	S0	(rst)
10	S3	S4	(din).(!rst)
11	S3	S1	(!din).(!rst)
12	S3	S0	(rst)
13	S4	S5	(!din).(!rst)
14	S4	S4	(din).(!rst)
15	S4	S0	(rst)
16	S5	S3	(din).(!rst)
17	S5	S1	(!din).(!rst)
18	S5	S0	(rst)

Output Video Link:

https://drive.google.com/file/d/1JQ8l7yAKydUXH3CkrdGmkK2xn6ZCkIfX/view?usp=drivesdk

Date:31/12/2021