INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

Department
of
Electronics & Communication Engineering

Embedded Logic Design(ECE270)

Dr. Sumit J Darak

Lab_4: Design and implement an overlapping sequence detector using Mealy FSM and verify its functionality

Mohammad Shariq
2020220

16-10-2021

OBJECTIVE:

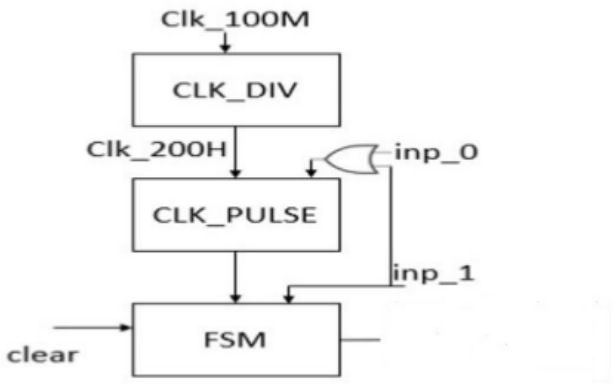
- ➢ 1. Design and implement an overlapping sequence detector using Mealy FSMand verify its functionality using testbench.
- ➢ 2. Design a clock pulse generator.
- ➢ 3. Test the functionality of the design on the Basys 3 Board (remote access) using VIO IP.

Topics to Explore:1) Mealy and Moore FSM implementation, 2) Use of 3 Always blocks, 3) Difference between overlapping and non-overlapping

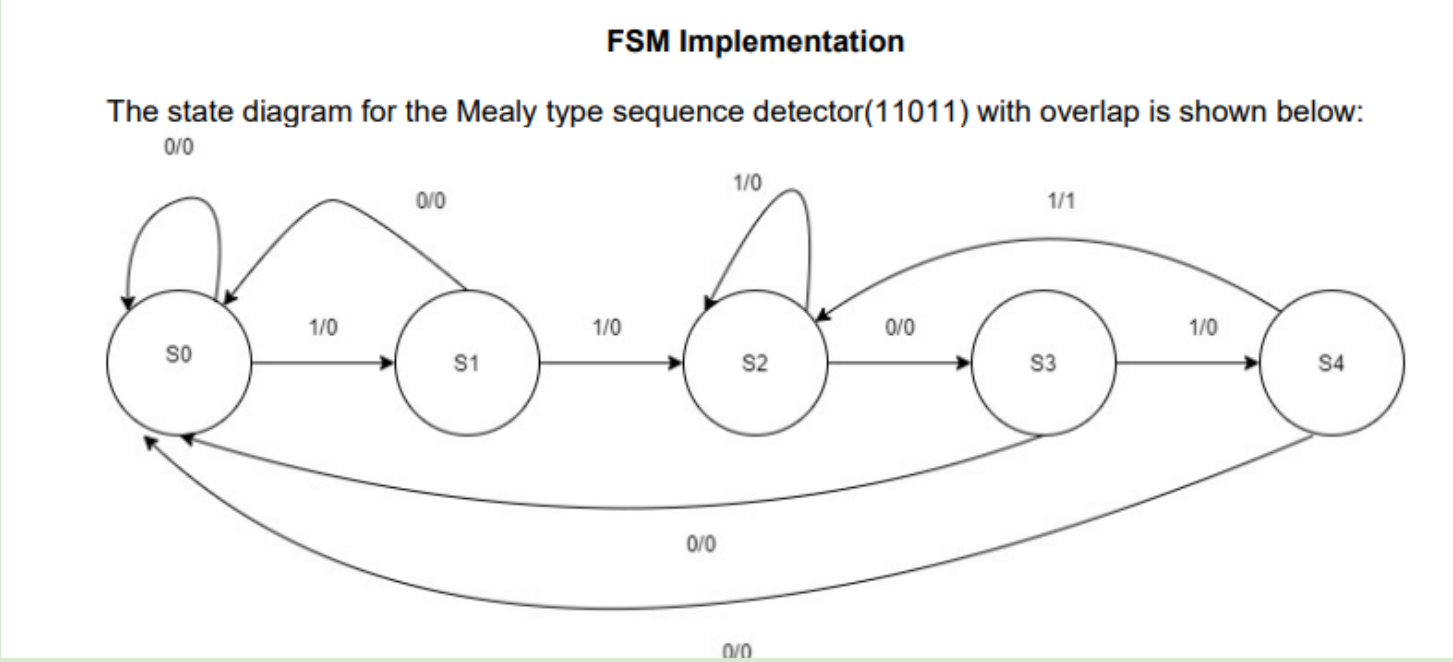
Theory:

- ## Clock pulse Generator

The first thing we need to consider is the **clock pulse generation** for the FSM. We will use a 200 Hz clock to generate a clock pulse, which transitions from 0 to 1 whenever any of the data push buttons is pressed. The following block diagram will give you an abstract view of the implementation(all the ports are not shown).



- ***FSM:***



Observations:

programme for functionality of FSM

PROJECT MANAGER - FSM

Project Summary | fsm_11011.v

D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sources_1/new/fsm_11011.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 13.10.2021 11:38:12
7  // Design Name:
8  // Module Name: fsm_11011
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module fsm_11011(
24
25     input clear,
26     input inp_1,
27     input input_pulse,
28     output reg out=0,
29     output reg [2:0] present_state
30    );
31
32    reg[2:0] next_state;
33    parameter S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011, S4=3'b100;  //parameter are defined so that we don't need to remember the
```

PROJECT MANAGER - FSM

Sources | Source File Properties

Project Summary | fsm_11011.v

D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sources_1/new/fsm_11011.v

```
31
32      reg[2:0] next_state;
33      parameter S0=3'b000,S1=3'b0001,S2=3'b010,S3=3'b011,S4=3'b100;  //parameter are defined so that we don't need to remember the
34                                                                     // also it makes the code more readable
35
36
37      always @(posedge input_pulse or posedge clear)
38       begin
39           if(clear ==1'b1)
40               present_state<=S0;
41           else
42               present_state<=next_state;
43
44       end
45
46    always @(*)
47          begin
48             next_state = present_state;
49             case (present_state)
50                  S0:if (inp_1==1'b1)
51                          next_state =S1;
52                     else
53                     next_state=S0;
54                  S1:if (inp_1==1'b1)
55                          next_state =S2;
56                     else
57                           next_state=S0;
58                  S2:if (inp_1==1'b0)
59                          next_state =S3;
60                     else
61                           next_state=S2;
62
63                  S3:if (inp_1==1'b1)
```

PROJECT MANAGER - FSM

Sources

Source File Properties

Project Summary | fsm_11011.v

D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sources_1/new/fsm_

```
54                  S1:if (inp_1==1'b1)
55                          next_state =S2;
56                     else
57                          next_state=S0;
58                  S2:if (inp_1==1'b0)
59                          next_state =S3;
60                     else
61                          next_state=S2;
62
63                  S3:if (inp_1==1'b1)
64                          next_state =S4;
65                     else
66                          next_state=S0;
67                  S4:if (inp_1==1'b1)
68                          next_state =S2;
69                     else
70                          next_state=S0;
71
72                  default : next_state=S0;
73             endcase
74         end
75
76    always @(posedge input_pulse)
77      begin
78              if(present_state==S4 && inp_1 == 1'b1)
79                      out<=1;
80              else
81                  out<=0;
82      end
83
84  endmodule
85
```

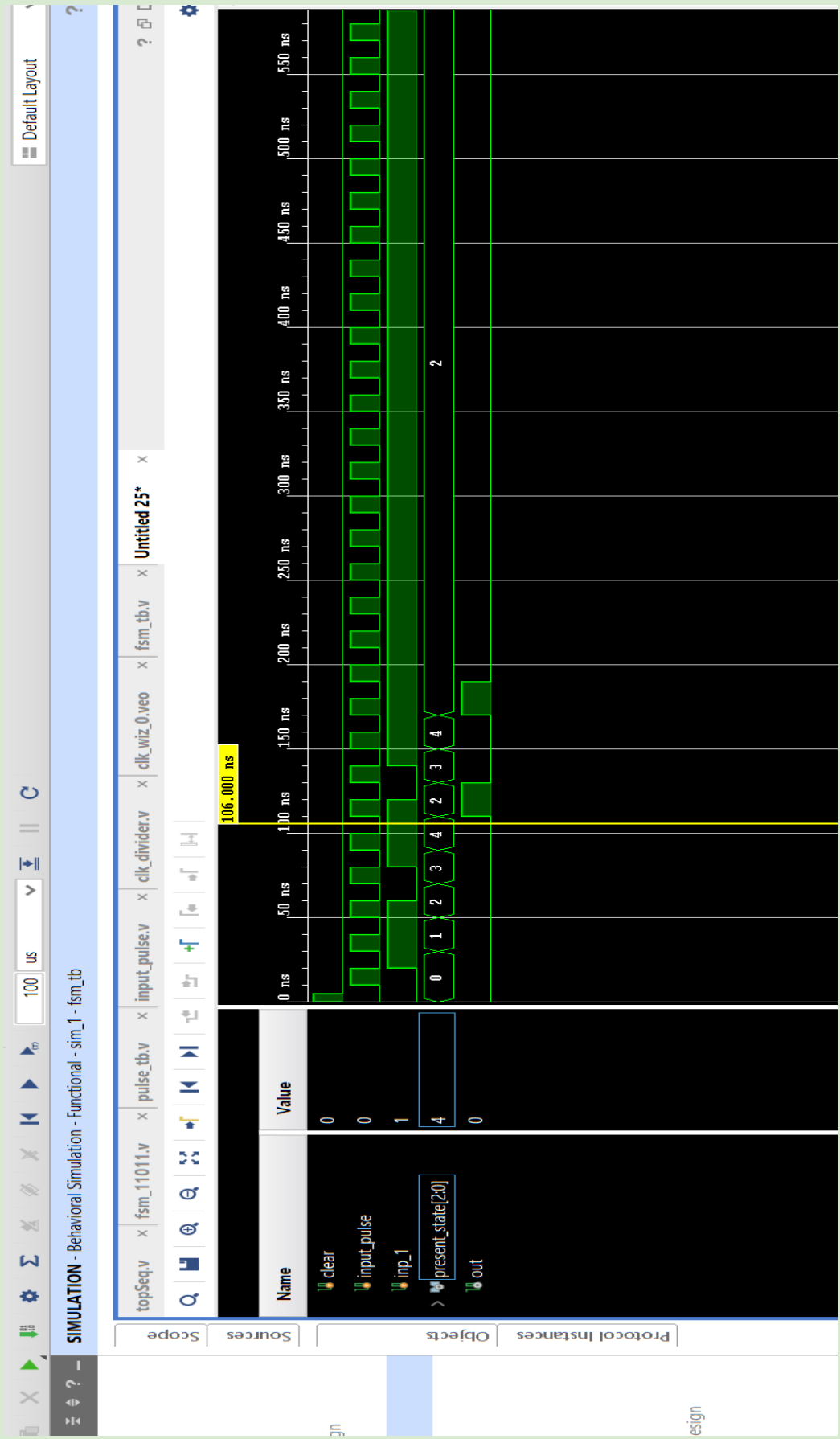
Testbench for FSM

fsm_tb.v

D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sim_1/new/fsm_tb.v

```
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module fsm_tb(
24
25      );
26
27      reg clear,input_pulse , inp_1;
28      wire [2:0] present_state;
29      wire out;
30
31      fsm_11011 tb1(.input_pulse(input_pulse),.clear(clear),.inp_1(inp_1),.out(out),.present_state(present_state));
32
33      initial begin
34          input_pulse<=1'b0;
35          clear<=1'b1;
36          inp_1<=1'b0;
37
38      end
39
40      initial begin
41
42      #5 clear=0;
43      @(posedge input_pulse) inp_1<=1;  // changing the input data at negative edge of the clock*
44      @(posedge input_pulse) inp_1<=1;
45      @(posedge input_pulse) inp_1<=0;
46      @(posedge input_pulse) inp_1<=1;
47      @(posedge input_pulse) inp_1<=1;  // pattern is completed
48      @(posedge input_pulse) inp_1<=0;
49      @(posedge input_pulse) inp_1<=1;
50      @(posedge input_pulse) inp_1<=1;   // overlaped pattern is completed again
51
52      end
53
54      always #10 input_pulse=~input_pulse;
55
56  endmodule
57
```

Results for automated testbench:



- ***program for functionality Pulse_Generaor:***

Project Summary | fsm_11011.v | pulse_tb.v | input_pulse.v

D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sources_1/new/input_pulse.v

```
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module input_pulse(
24      input clk_200H,
25      input inp_0, //7: ;iv, idput as 7
26      input inp_1, //7: plye .1Apst as 1
27      output input_pulse
28  );
29
30      reg Q= 0;
31      reg D = 0;
32
33      wire inp_pulse;
34
35      assign inp_pulse= inp_0 | inp_1;
36
37      always @(posedge clk_200H)
38      begin
39          Q <= D;
40      end
41
42      always @(*) begin
43          D= inp_pulse;
44      end
45      assign input_pulse=Q;
46
47  endmodule
48
49
```

- ***Testbench program for functionality Pulse_Generaor:***

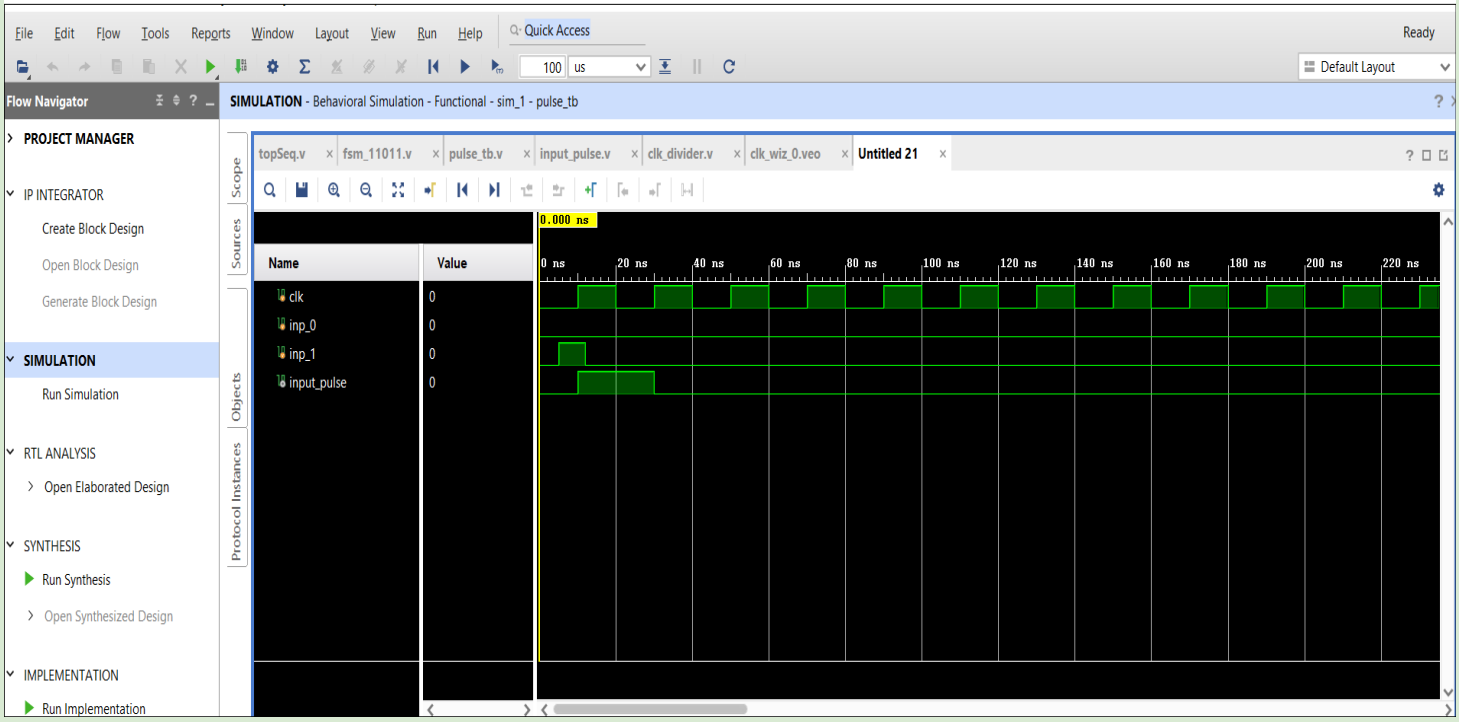
Default Layou

Project Summary | fsm_11011.v | pulse_tb.v | input_pulse.v

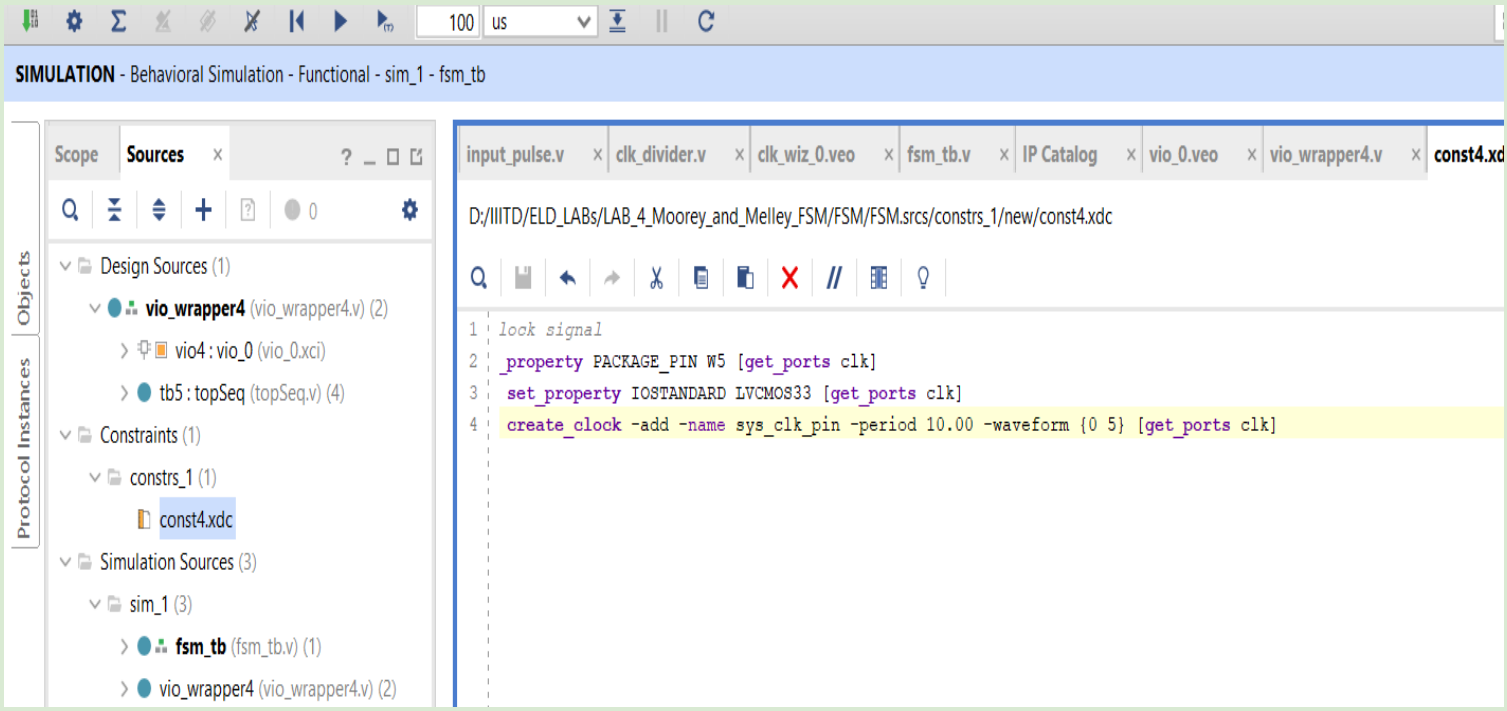
D:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sim_1/new/pulse_tb.v

```
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module pulse_tb(
24
25      );
26
27    reg clk, inp_0,inp_1;
28    wire input_pulse;
29
30      input_pulse tb0(.clk_200H(clk),.inp_0(inp_0) ,.inp_1(inp_1) , .input_pulse(input_pulse));
31
32      initial
33          begin
34              clk =0;
35              inp_0 = 0; inp_1= 0;
36          end
37
38      initial begin
39          #5 inp_0=0 ; inp_1 =1;
40          #7 inp_0=0; inp_1=0;
41
42          // for doing another test
43          //# inp_0=0; inp_1=1;
44          //  #inp_0=0; inp_1=0;
45      end
46
47      always #10 clk = ~clk;
48
49  endmodule
50
```

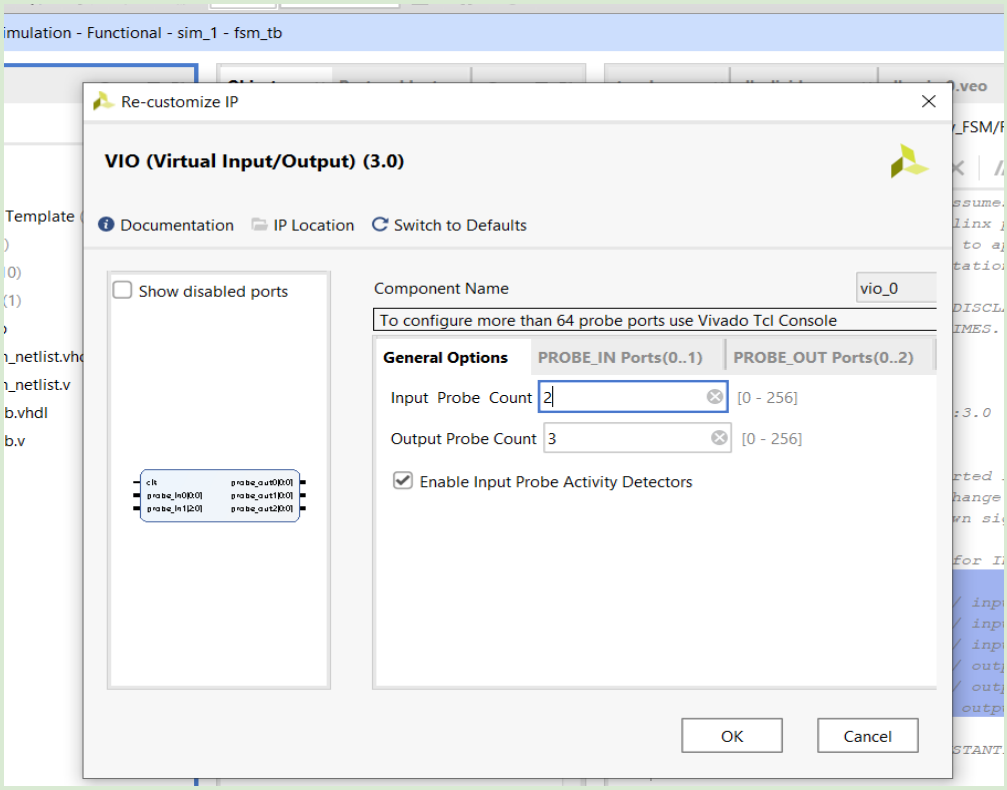
Test Result for Input Pulse:

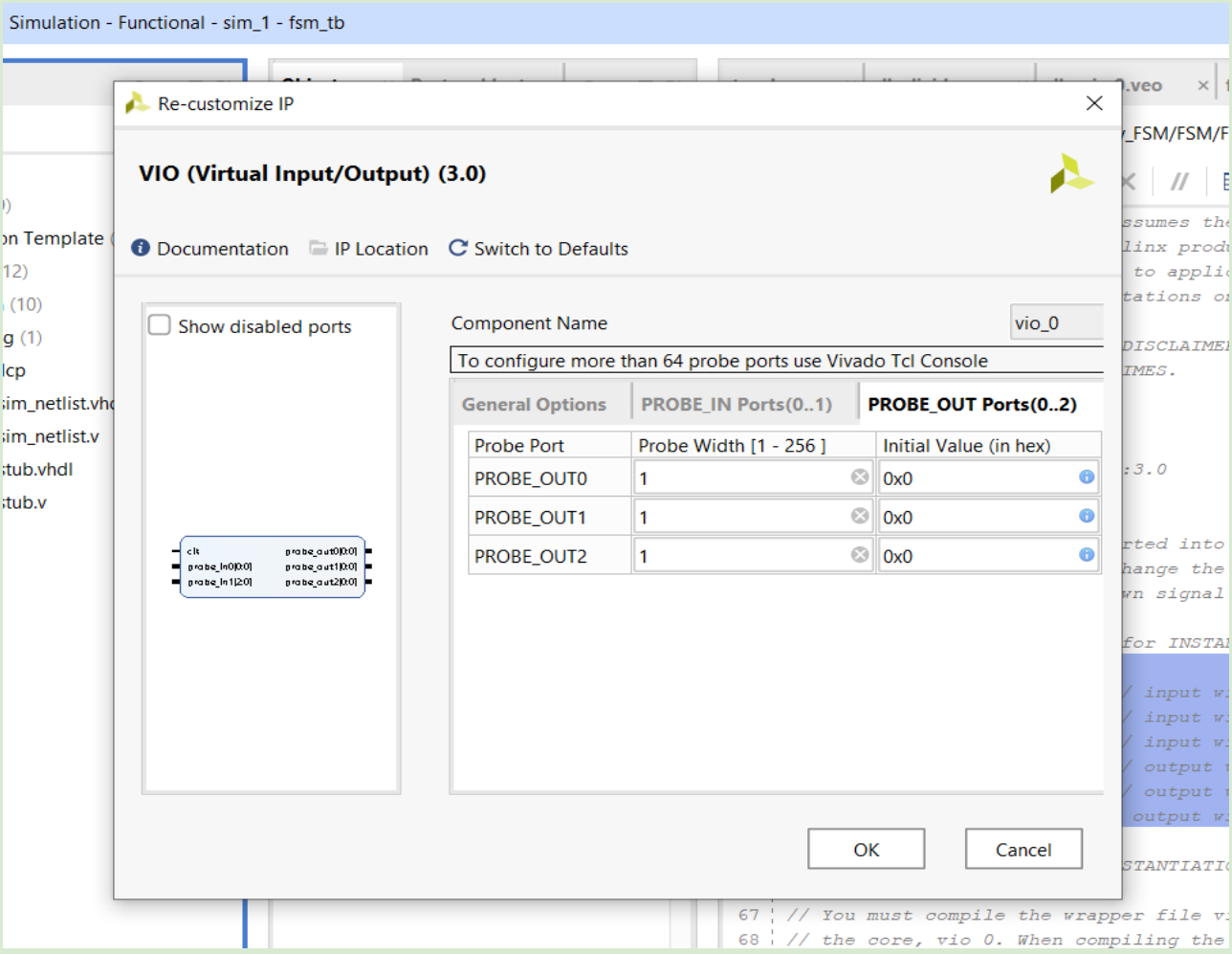


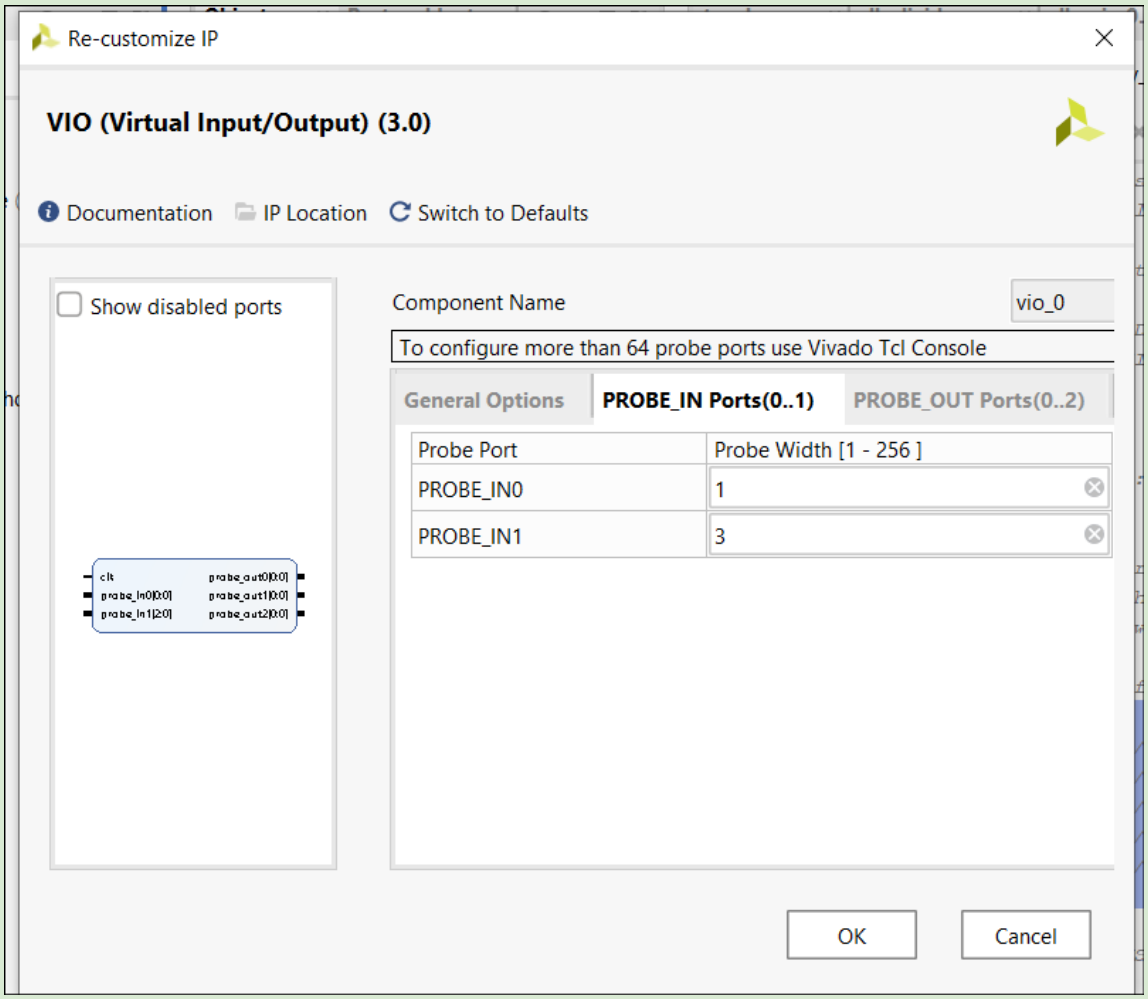
Constraints for the implementation and VIO module programme



VIO Wrapper Settings:





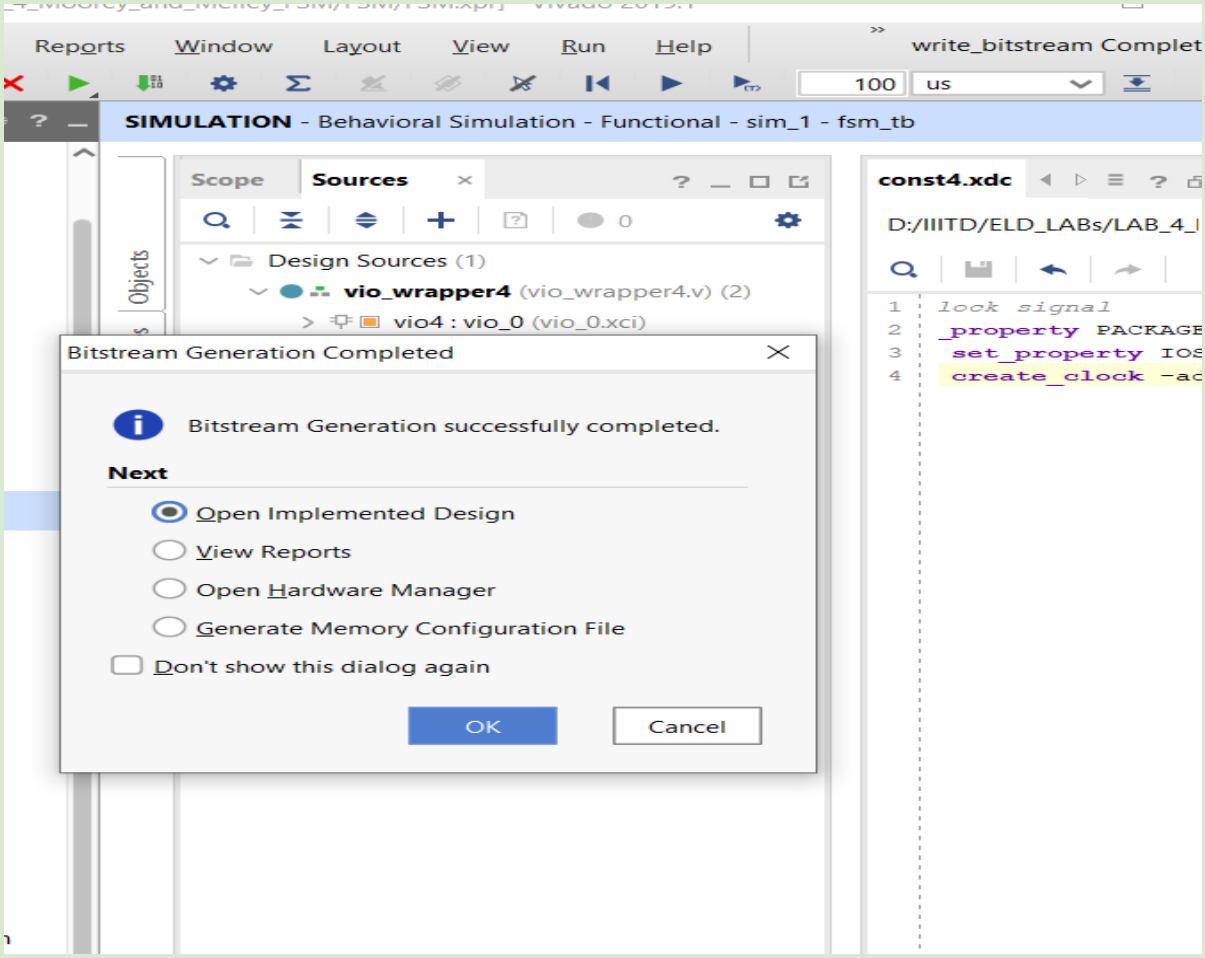


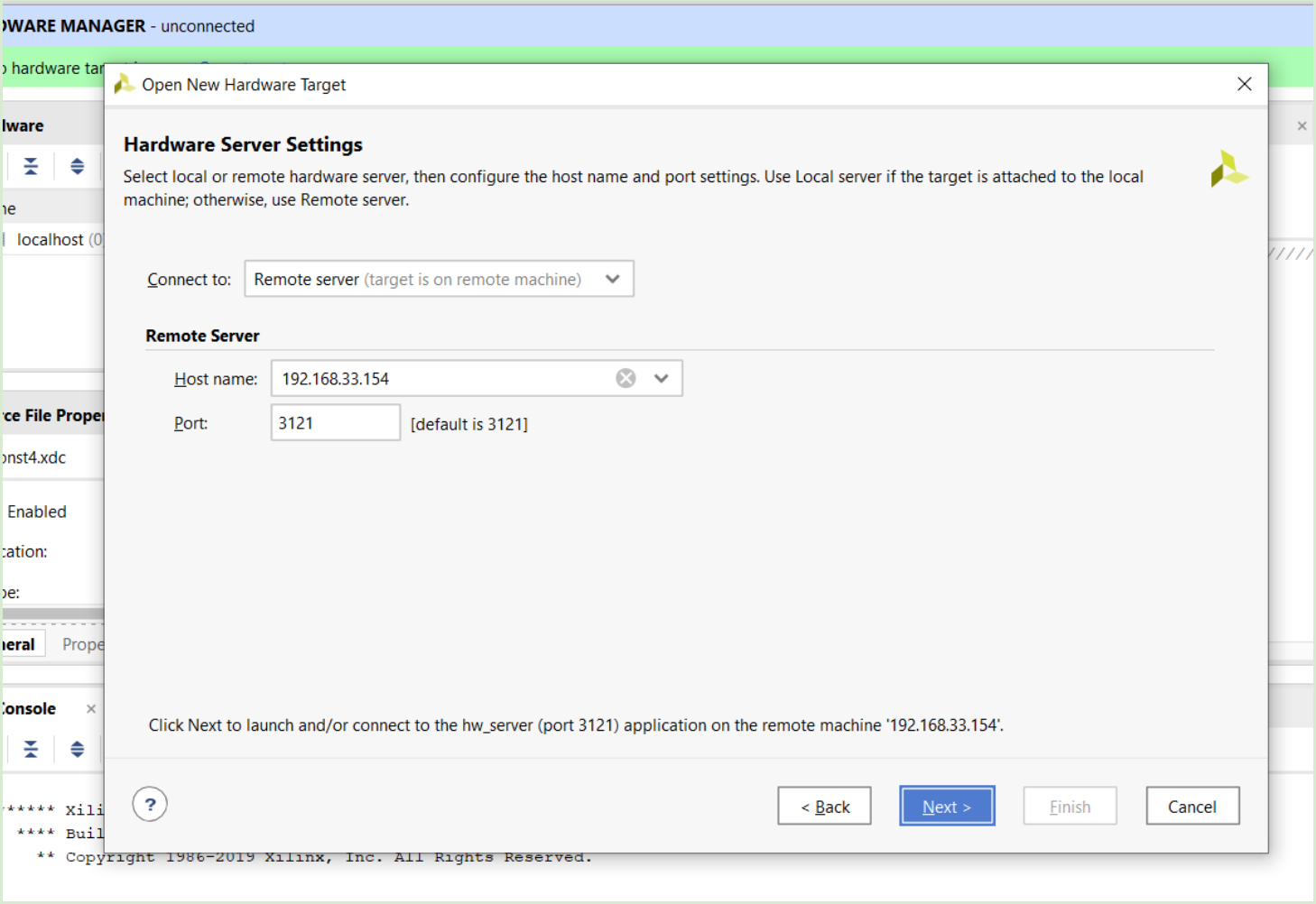
d:/IIITD/ELD_LABs/LAB_4_Moorey_and_Melley_FSM/FSM/FSM.srcs/sources_1/ip/vio_0/vio_0.veo

```
39 // Applications"). Customer assumes the sole risk and
40 // liability of any use of Xilinx products in Critical
41 // Applications, subject only to applicable laws and
42 // regulations governing limitations on product liability.
43 //
44 // THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
45 // PART OF THIS FILE AT ALL TIMES.
46 //
47 // DO NOT MODIFY THIS FILE.
48
49 // IP VLNV: xilinx.com:ip:vio:3.0
50 // IP Revision: 20
51
52 // The following must be inserted into your Verilog file for this
53 // core to be instantiated. Change the instance name and port connections
54 // (in parentheses) to your own signal names.
55
56 //----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
57 vio_0 your_instance_name (
58   .clk(clk),                  // input wire clk
59   .probe_in0(probe_in0),      // input wire [0 : 0] probe_in0
60   .probe_in1(probe_in1),      // input wire [2 : 0] probe_in1
61   .probe_out0(probe_out0),    // output wire [0 : 0] probe_out0
62   .probe_out1(probe_out1),    // output wire [0 : 0] probe_out1
63   .probe_out2(probe_out2)     // output wire [0 : 0] probe_out2
64 );
65 // INST_TAG_END ------ End INSTANTIATION Template ---------
66
67 // You must compile the wrapper file vio_0.v when simulating
68 // the core, vio_0. When compiling the wrapper file, be sure to
69 // reference the Verilog simulation library.
70
```

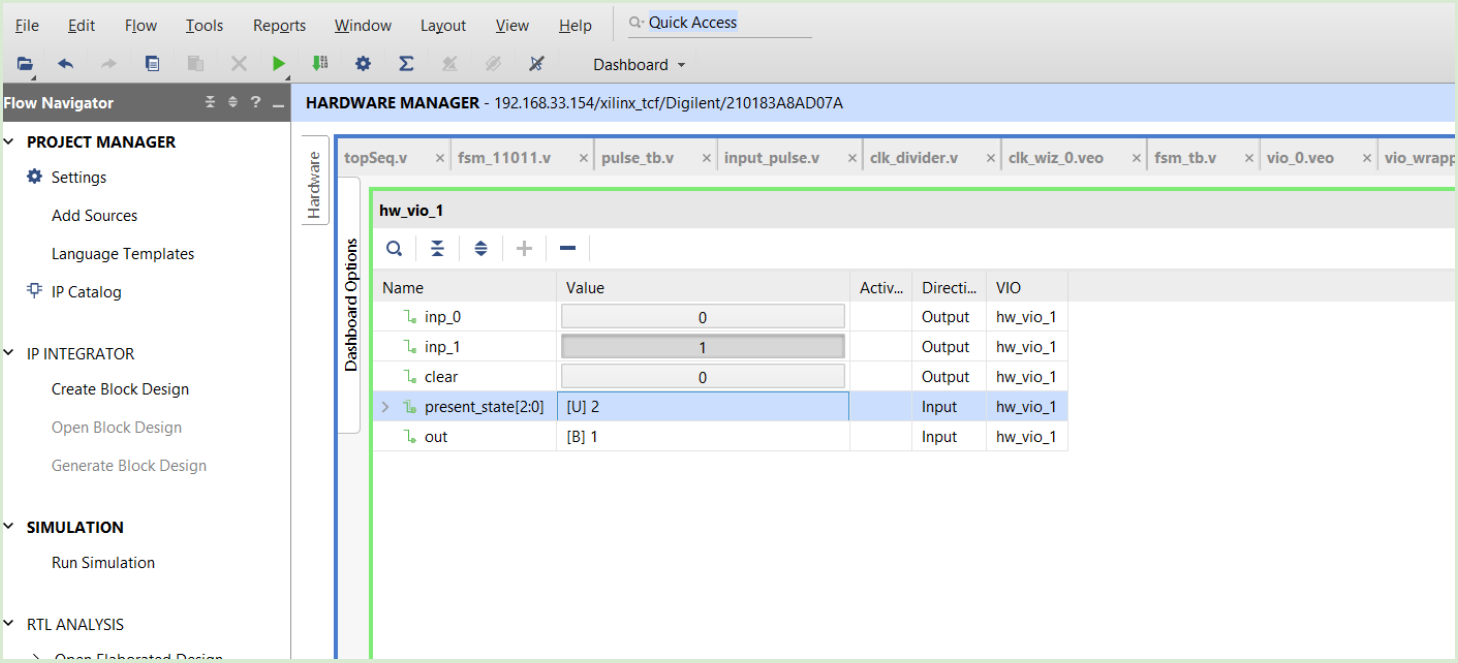
```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vio_wrapper4(
    input clk
    );

    wire clear, inp_0,inp_1,out;
    wire[2:0] present_state;

    vio_0 vio4(
  .clk(clk),                  // input wire clk
  .probe_in0(out),     // input wire [0 : 0] probe_in0
  .probe_in1(present_state),     // input wire [2 : 0] probe_in1
  .probe_out0(inp_0),  // output wire [0 : 0] probe_out0
  .probe_out1(inp_1),  // output wire [0 : 0] probe_out1
  .probe_out2(clear)  // output wire [0 : 0] probe_out2
 );


    topSeq tb5(.clk_100M(clk),.clear(clear),.input_0(inp_0),.input_1(inp_1),.out(out),.present_state(present_state));

endmodule

```

BitStream Generated Successfully:





Testing on BASYS3 Board using VIO IP



Conclusion:

Successfully tested functionality of an overlapping Sequence detector using Mealy FSM on BASYS3 board using VIO IP (Remote Access).