ELD LAB HOMEWORK 5

Aditya Gautam
2023043

- Demo Video
  - https://drive.google.com/file/d/1y3z8QRbyMl5KMELx9vd7N2Hw_RAMhs0t/view?usp=sharing

- Project Directory

```
vio_wrapper.v
|___ top_FSM.v
|      |___ cmt (ip)
|      |___ clk_div_rtl.v
|      |___ pb_clk.v
|      |___ FSM_moore_1111.v
|___ vio (ip)
```

- Code

1. vio_wrapper.v

```
module vio_wrapper (
    input Clk_100M
);

  wire detect;
  wire [2:0] FSM_state;
  wire clear;
  wire inp_0, inp_1, clear;

  top_FSM t1(
    .inp_1(inp_1),
    .Clk_100M(Clk_100M),
    .clear(clear),
    .detect(detect),
    .FSM_state(FSM_state)
  );

  vio_0 vio (
    .clk(Clk_100M),                // input wire clk
    .probe_in0(detect),    // input wire [0 : 0] probe_in0
    .probe_in1(FSM_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
  );


endmodule
```

repo/notes/sem3/ELD2024_LAB/lab5/lab5.srcs/sources_1/new/vio_wrapper.v

2. top_FSM.v

```
module top_FSM (
    input inp_0, inp_1,
    input Clk_100M,
    input clear,
    output detect,
    output [2:0] FSM_state
);

  wire Clk_8M, Clk_1Hz, inp_1, inp_0, clk_pb;

  // CMT INSTANTIATION
  clk_wiz_0 cmt (
      // Clock out ports
      .Clk_8M  (Clk_8M),   // output Clk_8M
      // Clock in ports
      .Clk_100M(Clk_100M)
  );  // input Clk_100M

  // FREQ DIVISION INSTANTIATION
  clk_div_rtl fd (
      .Clk_8M (Clk_8M),
      .Clk_1Hz(Clk_1Hz)
  );

  pb_clk pb (
      .Clk_1Hz(Clk_1Hz), .inp_0(inp_0), .inp_1(inp_1), .clk_pb(clk_pb)
  );

  // COUNTER INSTANTIATION
  FSM_moore_1101 fsm(
      .clk_pb(clk_pb), .clear(clear), .inp_1(inp_1), .detect(detect), .FSM_state(FSM_state)
  );

endmodule
```

repo/notes/sem3/ELD2024_LAB/lab5/lab5.srcs/sources_1/new/top_FSM.v

3. clk_div_rtl.v

```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clk_div_rtl (
    input  Clk_8M,
    output Clk_1Hz
);

  reg [22:0] Count_reg = 0;  // Initialization of FFs during FPGA configuration
  reg [22:0] Count_next;  // output of combinational circuit...can not be initialized
  always @(posedge Clk_8M) begin
    Count_reg <= Count_next;  // D-FF
  end

  always @(*)  // Comb. ckt to find out next state
    Count_next = Count_reg + 1;

  assign Clk_1Hz = Count_reg[22];

endmodule
```

repo/notes/sem3/ELD2024_LAB/lab5/lab5.srcs/sources_1/imports/Downloads/clk_div_rtl.v
Table Mode Enabled

4. pb_clk.v

```
// Design Name:
// Module Name: pb_clk
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module pb_clk (
    input Clk_1Hz,
    input inp_0,
    input inp_1,
    output reg clk_pb
  );
  assign inp_pulse = inp_0 | inp_1;

  always @ (posedge Clk_1Hz)
    begin
      clk_pb <= inp_pulse;
    end

endmodule
```

repo/notes/sem3/ELD2024_LAB/lab5/lab5.srcs/sources_1/new/pb_clk.v
Table Mode Enabled

5. FSM_moore_1101.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 08/27/2024 09:48:36 AM
// Design Name:
// Module Name: counter_8bit
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


// 1
module FSM_moore_1101 (
    input clk_pb,
    input clear,
    input inp_1,
    output reg detect,
    output [2:0] FSM_state
);

  // 2
  parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
  reg [2:0] present_state = S0;
  reg [2:0] next_state;

  // 3 STATE REGISTER
  always @(posedge clk_pb or posedge clear) begin
    if (clear)
      present_state <= S0;
    else
      present_state <= next_state;
  end

  // 4 Login Comb CKt
  always @(*) begin
    case (present_state)
      S0:
        if (inp_1 == 1)
          next_state = S1;
        else
          next_state = S0;
      S1:
        if (inp_1 == 1)
          next_state = S2;
        else
          next_state = S0;
      S2:
        if (inp_1 == 1)
          next_state = S3;
        else
          next_state = S0;
      S3:
        if (inp_1 == 1)
          next_state = S4;
        else
          next_state = S0;
      S4:
        if (inp_1 == 1)
          next_state = S4;
        else
          next_state = S0;
      default:
        next_state = S0;
    endcase
  end

  // 5 Output Comb Ckt
  always @(*) begin
    if (next_state == S4)
      detect = 1;
    else
      detect = 0;
  end

  assign FSM_state = present_state;

endmodule
```

- NOTE: I meant to name this main logic file as FSM_moore_1111.v but since there was not much to change in the main lab code, I continued in the same project and hence it is named FSM_moore_1101.v here.