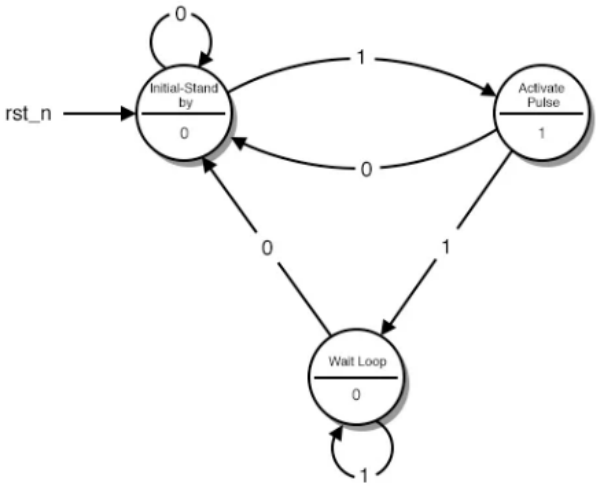
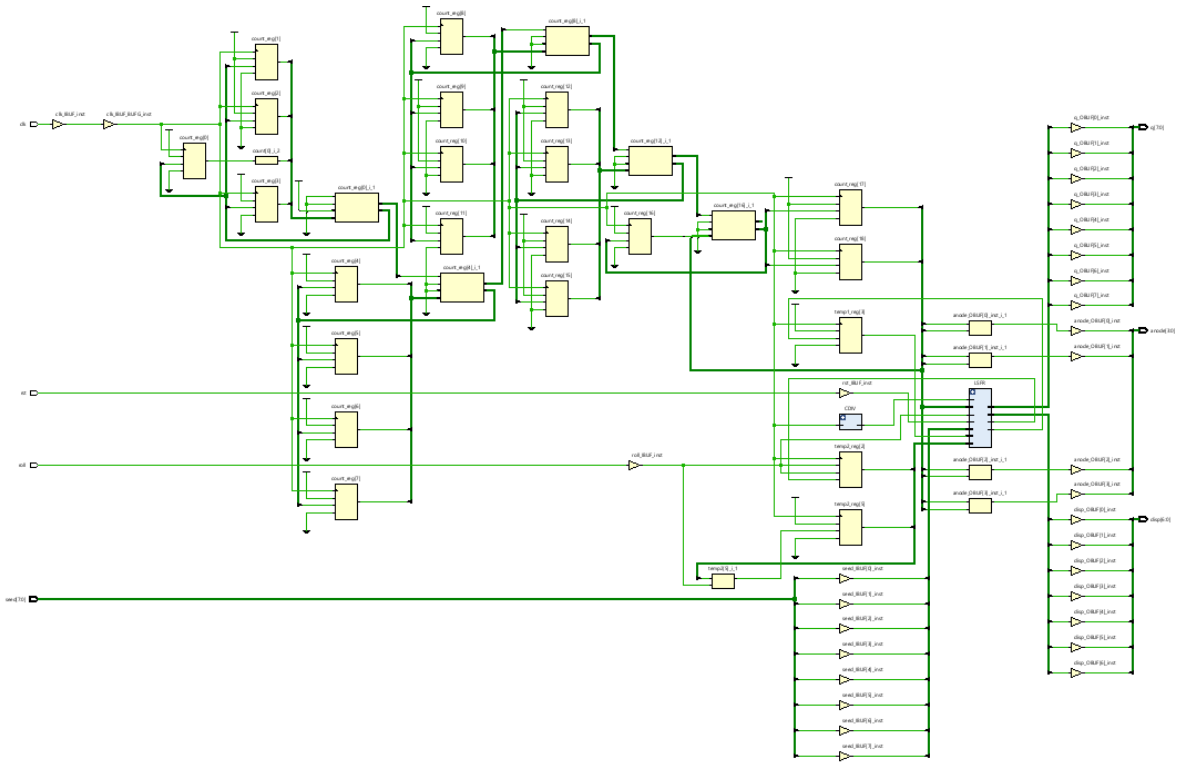
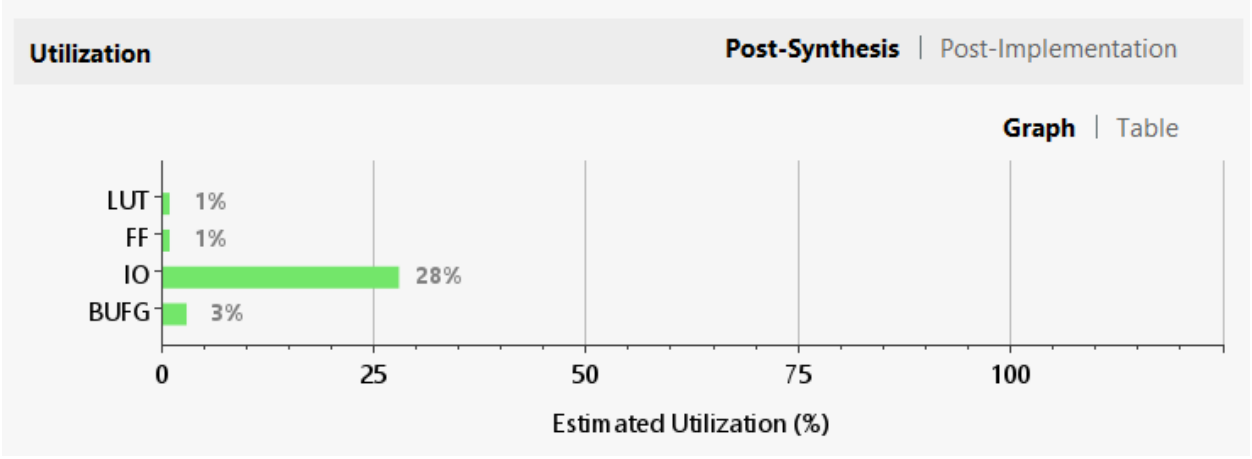
Vincent Han

Lab 5 Notebook

1. I had utilized the previous code from the last lab to be able to generate a state machine that plays a game when a button is pressed.
2. Tested each state and the inputs to get it to the next state.

3. 

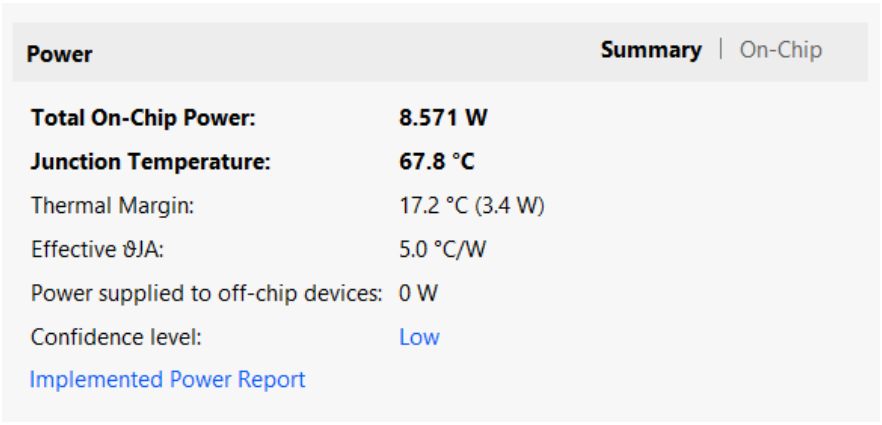
4. 

5. 

6. **Design Timing Summary**

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	315	Total Number of Endpoints:	315	Total Number of Endpoints:	NA

There are no user specified timing constraints.

7. 

8. Design goals were met, the state machine worked as operated and then the bitstream was correctly transferred to the board.
9. The testbench file is always confusing me with the test cases.
10. I would clean up the XDC file.

Source Code:

game_tb.v

```
module GameFSM_TB;
```

```
reg clk;
reg reset;
reg start;
reg roll;

wire [3:0] seg;

GameFSM dut (
  .clk(clk),
  .reset(reset),
  .start(start),
  .roll(roll),
  .seg(seg)
);

always begin
  clk = 1'b0;
  #5;
  clk = 1'b1;
  #5;
end

initial begin
  reset = 1'b1;
  start = 1'b0;
  roll = 1'b0;

  #10;
  reset = 1'b0;
  #10;

  start = 1'b1;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;

  start = 1'b1;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;

  start = 1'b1;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;

  start = 1'b1;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;
  roll = 1'b1;
  #10;
  roll = 1'b0;
  #10;
end
```

```
endmodule
```

Lab5_top.v

```
module lab4_top(
    output [7:0] q,
    output reg [6:0] disp,
    output reg [3:0] anode,
    input clk,
    input rst,
    input roll,
    input [7:0] seed
    );

    wire cout;
    wire [6:0] disp1;
    wire [6:0] disp2;

    initial
    begin
        disp <= 7'b1111111;
        anode <= 4'b1111;
    end

    clock_divider CDIV (cout, clk);
    lfsr LSFR (q, seed, rst, cout);
    Seven_segment digit1(disp1, q[3:0]);
    Seven_segment digit2(disp2, q[7:4]);

    localparam N = 19;
    reg [N-1:0] count = 0;
    reg [6:0] temp1 = 7'b1111111;
    reg [6:0] temp2 = 7'b1111111;
    always @ (posedge clk)
    begin
        count <= count + 1;
    end

    always @ (posedge clk)
    begin
        if(roll == 1)
        begin
            if(q[7:4] == q[3:0])
            begin
                temp1 <= 7'b1001111;
                temp2 <= 7'b1000001;
            end
            else
            begin
                temp1 <= 7'b1000000;
                temp2 <= 7'b1000111;
            end
        end
    end

    always @ (*)
    begin
        case(count[N-1:N-2])
            2'b00: begin
                disp <= disp1;
                anode <= 4'b1110;
            end
            2'b01: begin
                disp <= disp2;
                anode <= 4'b1101;
            end
```

```
            2'b10: begin
                disp <= temp1;
                anode <= 4'b1011;
            end
            2'b11: begin
                disp <= temp2;
                anode <= 4'b0111;
            end
        endcase
    end

endmodule
```