Experiment 2 - Sequential Synthesis and FPGA Device Programming

Mohammad Ghareh Hasanloo, 810198461

Danial Saeedi, 810198571

Abstract— **This is the report of experiment 2 and in this experiment, we have to get familiar with the concepts of state machines that are mostly used for controllers. The experiment consists of two parts. First a serial transmitter circuit is designed and then DE1 development board is used for our design implementation.**

Keywords— **State Machine, Sequence Detectors, Huffman Coding Style, Synthesis, Design Simulation, FPGA Programming, Onepulser, Seven Segment Display**

1.1 Onepulser

Onepulser is used for recognizing when the button is pushed. It makes output one for one clock, no matter how long the button is being pushed.

```
1   `timescale 1ns/1ns
2   module one_pulse(clk, clkPB, single_pulse);
3           input clk;
4           input clkPB;
5           output reg single_pulse;
6
7           logic [1:0] ps,ns;
8
9           always @(ps,clkPB) begin
10                  ns = 2'b0;
11                  case (ps)
12                          2'b00: ns = clkPB ? 2'b01 : 2'b00;
13                          2'b01: ns = 2'b10;
14                          2'b10: ns = clkPB ? 2'b10 : 2'b00;
15                          default: ns = 2'b00;
16                  endcase
17          end
18          always @(posedge clk) begin
19                  ps <= ns;
20          end
21
22          assign single_pulse = ps == 2'b01 ? 1'b1 : 1'b0;
23
24  endmodule
```

Fig. 1 Onepulser Verilog Description

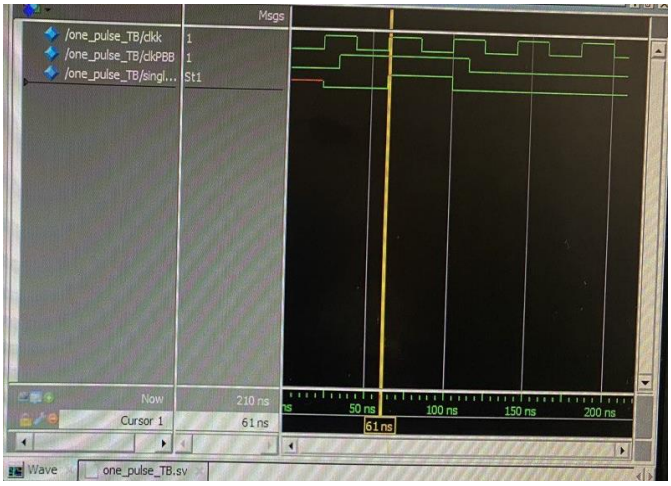

Fig. 2 Waveform of Onepulser Test Bench

1.2 Orthogonal Finite State Machine

A Moore state machine is designed with a counter. Once detector detects the sequence of 1011 on its input, the counter will be enable to count for 10 clock cycles and it counts when the button is pushed.

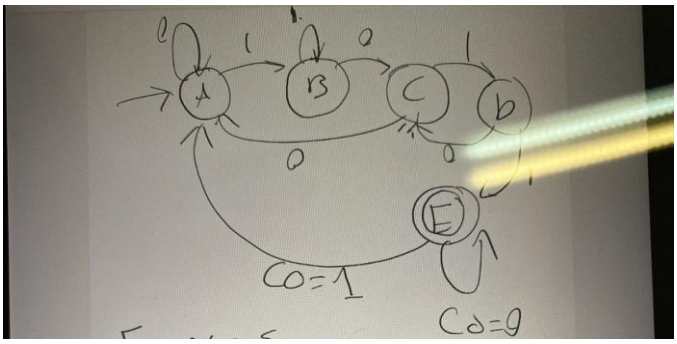

Fig. 3 State Diagram of the Sequence Detector for 1011

```
`timescale 1ns/1ns
module counter(clk,ci, clk_EN, inc_cnt,rst_cnt,co,count_out);
        input clk, clk_EN, inc_cnt,rst_cnt;
        input ci;
        output reg co;
        output reg[3:0] count_out;

        logic [1:0] ps,ns;

        always @(posedge clk_EN,posedge rst_cnt) begin
                if (rst_cnt)
                        count_out <= 4'b0110;
                else begin
                        if (inc_cnt)
                                count_out <= ci ? (count_out + 1'b1) : count_out;
                end
        end

        assign co = (inc_cnt) ? &{count_out,ci} : 1'b0;

endmodule
```

Fig. 4 Counter Verilog Description

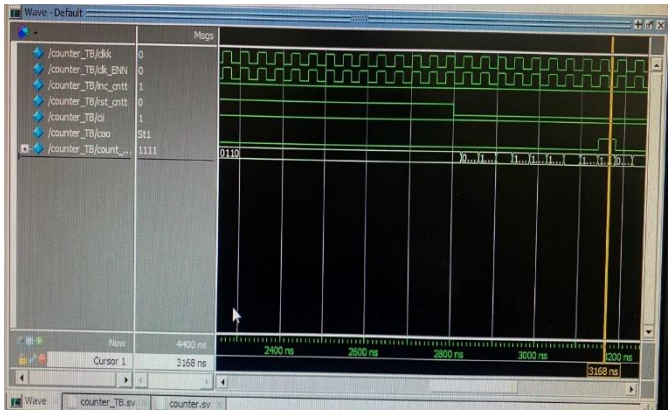

Fig. 5 Waveform of Counter Test Bench

```
`timescale 1ns/1ns
module sequence_detector_1011(clk,rst,serIn,co,serOut,serOutValid,inc_cnt,rst_cnt);
        input clk,rst,serIn,co;
        output reg serOut,serOutValid,inc_cnt;
        output reg rst_cnt;

        logic [2:0] ps,ns;

        always@(ps, serIn, co) begin
                {serOut, serOutValid, rst_cnt, inc_cnt} = 4'b0000;
                case (ps)
                        3'b000: begin rst_cnt = 1'b1; ns = serIn ? 3'b001 : 3'b000; end
                        3'b001: ns = serIn ? 3'b001 : 3'b010;
                        3'b010: ns = serIn ? 3'b011 : 3'b000;
                        3'b011: ns = serIn ? 3'b100 : 3'b010;
                        3'b100: begin ns = co ? 3'b000 : 3'b100; serOut = serIn; serOutValid = 1'b1; inc_cnt = 1'b1; end
                        default: ns = 3'b000;
                endcase
        end

        always @(posedge clk,posedge rst) begin
                if (rst)
                        ps <= 3'b000;
                else
                        ps <= ns;
        end
endmodule
```

Fig. 6 1011 Detector Verilog Description

```
module OFSM(input clkPB, clk, serIn, rst, output serOut, serOutValid, output[6:0] seven_seg);
  wire co, inc_cnt, rst_cnt;


  wire clk_EN;

  wire [3:0] Count_out;

  hexto7segment hexo(Count_out,seven_seg);
  one_pulse UUT(clk,clkPB,clk_EN);
  counter cnt (clk, 1'b1, clk_EN, inc_cnt, rst_cnt, co, Count_out);
  sequence_detector_1011 sd (clk_EN,rst, serIn, co, serOut, serOutValid, inc_cnt, rst_cnt);

endmodule
```

Fig. 7 Top-level Verilog Description of the Transmitter Circuit

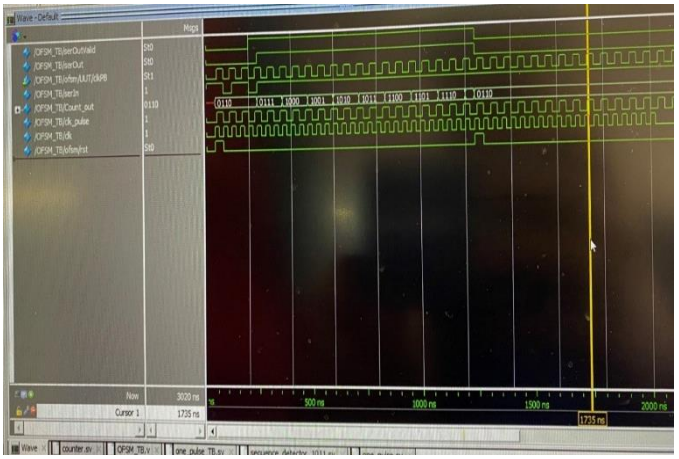

Fig. 8 Waveform of Top-level Verilog Description of the Transmitter Circuit

1.3 Seven Segment Display

Each seven segment receives a 4-bit input and displays the HEX value on its 7-bit output.

```
module hexto7segment(input  [3:0]x, output reg [6:0]z);
always @*
    case (x)
    4'b0000 :       //Hexadecimal 0
    z = 7'b1000000;
    4'b0001 :           //Hexadecimal 1
    z = 7'b1111001  ;
    4'b0010 :       // Hexadecimal 2
    z = 7'b0100100 ;
    4'b0011 :       // Hexadecimal 3
    z = 7'b0110000 ;
    4'b0100 :       // Hexadecimal 4
    z = 7'b0011001 ;
    4'b0101 :       // Hexadecimal 5
    z = 7'b0010010 ;
    4'b0110 :       // Hexadecimal 6
    z = 7'b0000010 ;
    4'b0111 :       // Hexadecimal 7
    z = 7'b1111000;
    4'b1000 :           //Hexadecimal 8
    z = 7'b0000000;
    4'b1001 :           //Hexadecimal 9
    z = 7'b0010000 ;
    4'b1010 :       // Hexadecimal A
    z = 7'b0001000 ;
    4'b1011 :       // Hexadecimal B
    z = 7'b0000011;
    4'b1100 :       // Hexadecimal C
    z = 7'b1000110 ;
    4'b1101 :       // Hexadecimal D
    z = 7'b0100001 ;
    4'b1110 :       // Hexadecimal E
    z = 7'b0000110 ;
    4'b1111 :       // Hexadecimal F
    z = 7'b0001110 ;
endcase

endmodule
```

Fig. 9 Verilog Description of the Seven Segment Display Module

2 Serial Transmitter Implementation

In this part, the result is described by followed photos step by step.

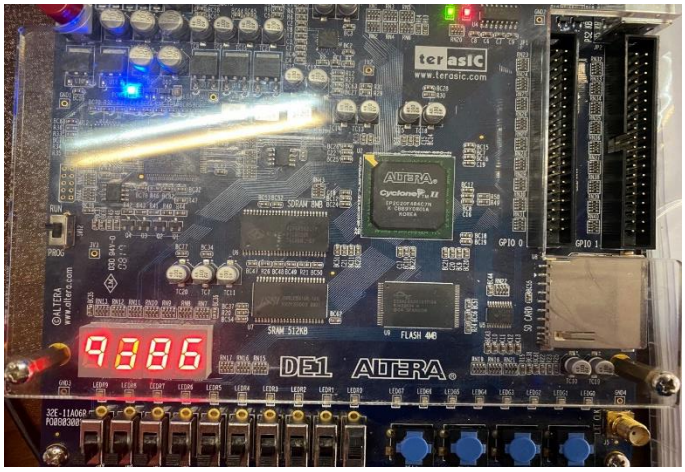

Fig. 10 Photo after Reseting

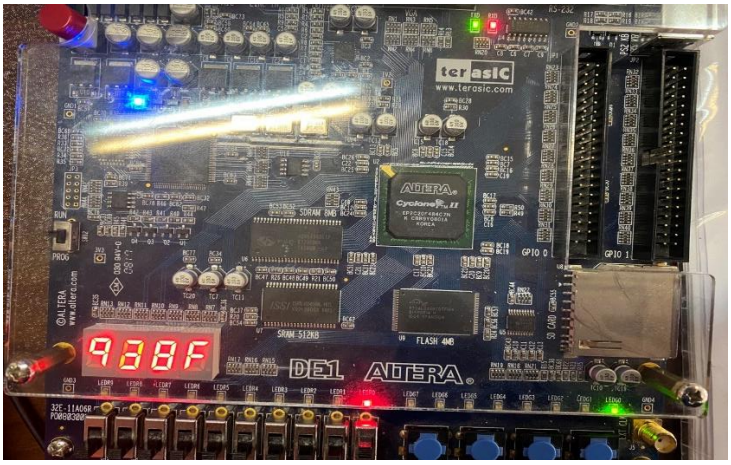

Fig. 12 Photo after Counting for 10 Times

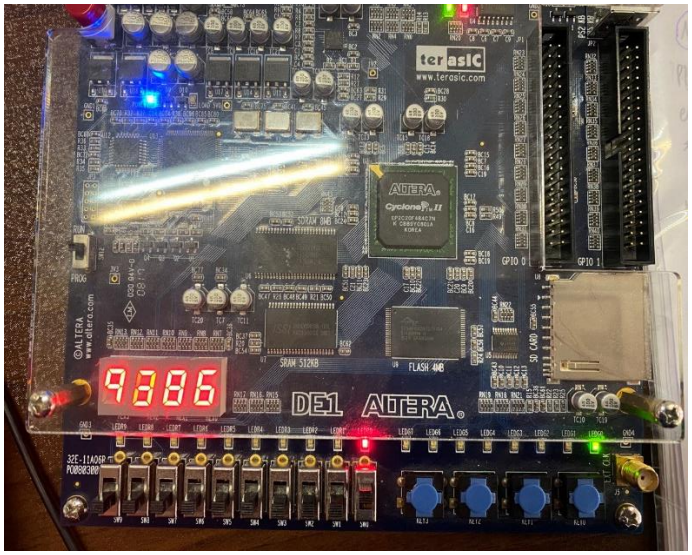

Fig. 11 Photo after Detecting 1011

First, there are two LEDs for serOut and serOutValid to show when each of them is one. When 1011 is detected, as you can see in picture above, two LEDs red and green are turned on. In this state, the serOutvalid and serOut becomes one and remains one for next 10 times the button pushing.

Next, as you can see in photo above, the counter starts to count from 6 to f. Then, the counter resets and again the detector waits for detecting 1011 on its input.