Experiment 2 – Sequential Synthesis and FPGA Programming

Mehdi Jamalkhah, 810100111

Mobina Mehrazar, 810100216

***Abstract*— This document is a report for experiment #2 of Digital Logic Design Laboratory at ECE department, University of Tehran. The purpose of this experiment is to bridge the gap between programming software and programming hardware through designing and executing FPGA application.**

***Keywords*— Include at least 5 keywords or phrases**

I. Introduction

The goal of this experiment is to introduce the concepts of state machines that are mostly used for controllers. The second goal is to get familiar with FPGA devices and implementation.

II. Serial Transmitter

A serial transmitter circuit is going to be designed that searches for a start sequence of 110101 on the serIn input and initiates transmission its serIn on its serOut. Upon detection of the start sequence, serOutValid is asserted, and the circuit transmits its serIn on its serOut for the next 10 clock cycles. After the entire 10 bits are transmitted, the circuit returns to its initial state, where it searches for the start sequence.

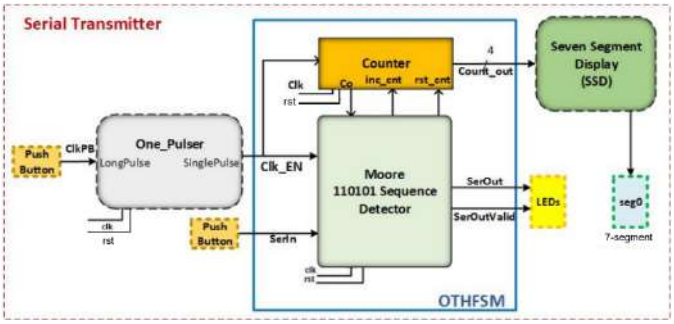

Fig. 1 Serial transmitter

A. Onepulser

If the clk signal in a Verilog code is implemented using a push button that is manually triggered by a human, the circuit's behaviour will depend on the timing of the button presses and releases.

For example, if the circuit should receive a 010 sequence, the button must be pressed and then released after a clock cycle (i.e. 0.02 seconds), which may be impossible.

A One-Pulser is a solution to this problem which occurs in circuits where the input is received through a push button key.

The One-Pulser circuit is a simple digital circuit that generates a single pulse that synchronized with the system clock. By connecting the push button input to the One-Pulser circuit, we can ensure that the circuit generates a consistent clock pulse every time the button is pressed, regardless of the timing or duration of the button press.

The output of the One-Pulser circuit can be used for controlling the clock when the circuit is implemented on an FPGA board, by being connected to the clock enable input (clkEn) of the sequence detector and the counter.

1) The Verilog description of the One-Pulser module is shown in Fig. 2.

```
Ln#
1   `timescale 1ns/1ns
2
3   module OnePulser(clk, rst, longPulse, singlePulse);
4           input clk, rst, longPulse;
5           output singlePulse;
6           reg singlePulse;
7           reg generated;
8
9           always @(posedge clk, posedge rst) begin
10                  if (rst) begin
11                          singlePulse = 1'b0;
12                          generated = 1'b0;
13                  end
14
15                  else begin
16                          if (singlePulse == 1'b1) begin
17                                  singlePulse = 1'b0;
18                          end
19
20                          else if (~longPulse && ~generated) begin
21                                  singlePulse = 1'b1;
22                                  generated = 1'b1;
23                          end
24
25                          if (longPulse && generated) begin
26                                  generated = 1'b0;
27                          end
28                  end
29          end
30
31  endmodule
```

Fig. 2 One-Pulser Verilog description

2) The design was tested in ModelSim and shown in Fig. 3.

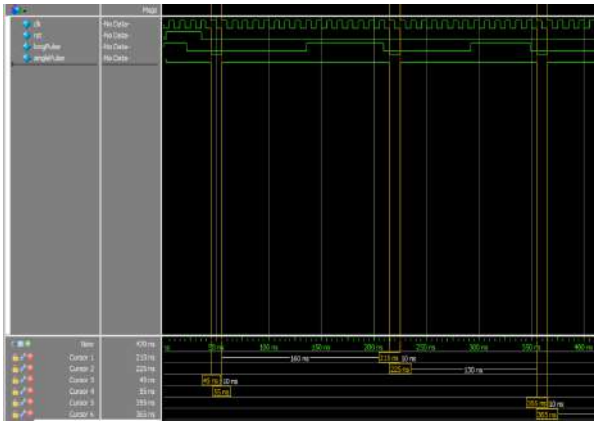

Fig. 3 One-Pulser simulation: the circuit receives a signal that the button has been pushed and asserts single pulse signal (at cursor 1, 3, and 5). After a clock cycle, it returns to zero (at cursor 2, 4, and 6) to create a complete single pulse. (Note: the long pulse is active low and the single pulse is active high)

B. Orthogonal Finite State Machine

An orthogonal finite state machine consists of a Moore state machine and a counter. When the clock Enable push-button is pressed, the Moore sequence detector checks its input and decides which state to transit. The sequence detector waits for the sequence of 110101 on its serIn input, and once received, the serOutValid output becomes one. The push button is pressed for every input that the state machine receives, after that the counter is enabled and counts for the next 10 consecutive clock cycles. During all these times, the serOutValid signal remains one.

1) The state diagram of the sequence detector is shown in Fig. 4.

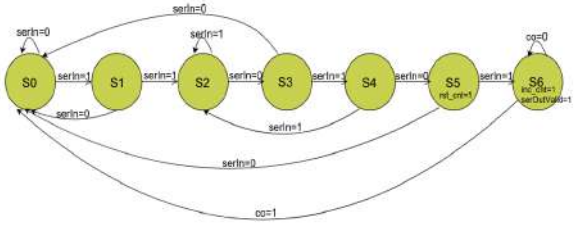

Fig. 4 110101 sequence detector state machine

2) The top-level Verilog description of the transmitter circuit is shown in Fig. 5 and Fig. 6. This top-level description includes the Verilog description of the sequence detector and the Verilog description of the counter, which are interfaced together to form the complete transmitter circuit.

```
1  `timescale 1ns/1ns
2
3  `define S0 3'b000
4  `define S1 3'b001
5  `define S2 3'b010
6  `define S3 3'b011
7  `define S4 3'b100
8  `define S5 3'b101
9  `define S6 3'b110
10
11 module SequenceDetector(clk, rst, serIn, clkEn, countOut, serOut, serOutValid);
12         input clk, rst, serIn, clkEn;
13         output [3:0] countOut;
14         output serOut, serOutValid;
15
16         reg serOutValid;
17         reg rstCnt, incCnt;
18         reg [2:0] ps, ns;
19         reg [3:0] countOut;
20         wire co;
21
22         always @(ps, co, serIn) begin
23                 ns = `S0;
24                 case (ps)
25                         `S0 : ns = serIn ? `S1 : `S0;
26                         `S1 : ns = serIn ? `S2 : `S0;
27                         `S2 : ns = serIn ? `S2 : `S3;
28                         `S3 : ns = serIn ? `S4 : `S0;
29                         `S4 : ns = serIn ? `S2 : `S5;
30                         `S5 : ns = serIn ? `S6 : `S0;
31                         `S6 : ns = co ? `S0 : `S6;
32                        default: ns = `S0;
33                 endcase
34         end
```

Fig. 5 Orthogonal finite state machine Verilog description (part 1)

```
35
36         always @(ps) begin
37                 {rstCnt, incCnt, serOutValid} = 3'b0;
38                 case (ps)
39                         `S0 : ;
40                         `S1 : ;
41                         `S2 : ;
42                         `S3 : ;
43                         `S4 : ;
44                         `S5 : rstCnt = 1'b1;
45                         `S6 : {incCnt, serOutValid} = 2'b11;
46                        default: {rstCnt, incCnt, serOutValid} = 3'b0;
47                 endcase
48         end
49
50         always @(posedge clk, posedge rst) begin
51                 if (rst) begin
52                         ps <= `S0;
53                 end
54
55                 else if (clkEn) begin
56                         ps <= ns;
57                 end
58         end
59
60         always @(posedge clk, posedge rst) begin
61                 if (rst) begin
62                         countOut = 4'b0;
63                 end
64
65                 else if (clkEn) begin
66                         if (rstCnt) begin
67                                 countOut = 4'b0110;
68                         end
69
70                         else if (incCnt) begin
71                                 countOut = countOut + 1;
72                         end
73                 end
74         end
75
76         assign co = &countOut;
77         assign serOut = serIn;
78 endmodule
```

Fig. 6 Orthogonal finite state machine Verilog description (part 2)

3) The result of simulation in ModelSim is shown in Fig. 7.

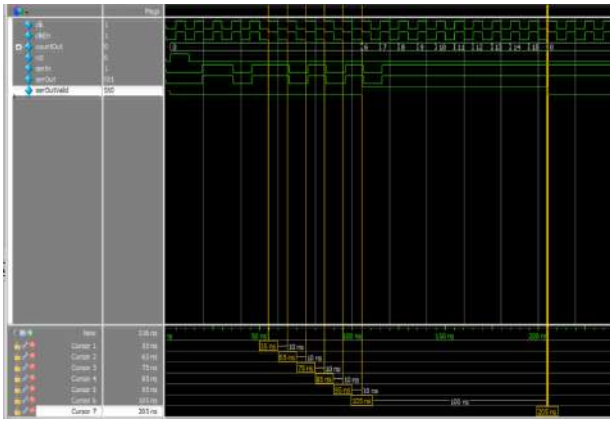

Fig. 7 Orthogonal finite state machine simulation: the state machine received 1, 1, 0, 1, 0, and 1 at cursor 1, 2, 3, 4, 5, and 6, respectively. Once the entire sequence of 110101 is detected the serOutValid signal becomes one (at cursor 6, same time as the last input is received) and after 10 clock cycles returns to zero (at cursor 7).

C. Seven Segment Display

The output of the counter will be displayed on the seven-segment display of the FPGA board. Each seven-segment receives a four-bit input and displays the hexadecimal value on its 7-bit output (Fig. 8).

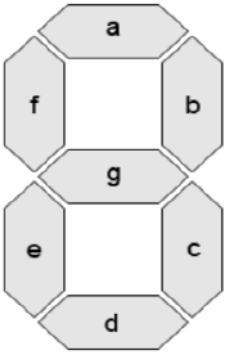

Fig. 8 Seven segment display

TABLE I
SEVEN SEGMENT DISPLAY TRUTH TABLE

decimal	B3 B2 B1 B0	A B C D E F G
0	0000	0000001
1	0001	1001111
2	0010	0010010
3	0011	0000110
4	0100	1001100
5	0101	0100100
6	0110	0100000
7	0111	0001111
8	1000	0000000
9	1001	0000100
10	1010	0001000
11	1011	1100000
12	1100	0110001
13	1101	1000010
14	1110	0110000
15	1111	0111000

1) Verilog description of the seven-segment display module is shown in Fig. 9 which is written based on table 1.

```
module SSD(dataIn, seg0);
        input [3:0] dataIn;
        output [6:0] seg0;
        reg [6:0] seg0;

        always @(dataIn) begin
                seg0 = 7'b0;
                case(dataIn)
                        4'd0: {seg0[6]} = 1'b1;
                        4'd1: {seg0 [6:3], seg0[0]} = 5'b1_1111;
                        4'd2: {seg0[5], seg0[2]} = 2'b11;
                        4'd3: {seg0[5:4]} = 2'b11;
                        4'd4: {seg0[4:3], seg0[0]} = 3'b111;
                        4'd5: {seg0[4], seg0[1]} = 2'b11;
                        4'd6: {seg0[1]} = 1'b1;
                        4'd7: {seg0[6:3]} = 4'b1111;
                        4'd8: ;
                        4'd9: {seg0[4]} = 1'b1;
                        4'd10: {seg0[3]} = 1'b1;
                        4'd11: {seg0[1:0]} = 2'b11;
                        4'd12: {seg0[2:1], seg0[6]} = 3'b111;
                        4'd13: {seg0[0], seg0[5]} = 2'b11;
                        4'd14: {seg0[2:1]} = 2'b11;
                        4'd15: {seg0[3:1]} = 3'b111;
                        default: seg0 = 7'b0;
                endcase
        end

endmodule
```

Fig. 9 Seven segment display Verilog description

2) The result of simulation in ModelSim is shown in Fig. 7.

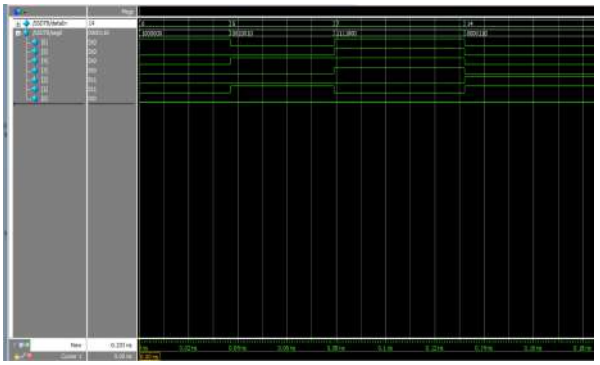

Fig. 10 Seven segment display simulation

III. SERIAL TRANSMITTER IMPLEMENTATION

A top-level Verilog description was created for the implementation of a design using the DE1 development board. The description includes all three parts, namely the One-Pulser, sequence detector, and seven-segment display modules, connected together (Fig. 11).

```
`timescale 1ns/1ns

module SerialTransmitter(clk, rst, clkPB, serIn, seg0, serOut, serOutValid);
        input clk, rst, clkPB, serIn;
        output [6:0] seg0;
        output serOut, serOutValid;

        wire clkEn;
        wire [3:0] countOut;

        OnePulser onePulser (.clk(clk), .rst(rst), .longPulse(clkPB), .singlePulse(clkEn));
        SequenceDetector sequenceDetector (.clk(clk), .rst(rst), .serIn(serIn), .clkEn(clkEn),
                                           .countOut(countOut), .serOut(serOut), .serOutValid(serOutValid));
        SSD ssd (.dataIn(countOut), .seg0(seg0));


endmodule
```

Fig. 11 Serial transmitter Verilog description

The result of Serial transmitter simulation in ModelSim is shown below:

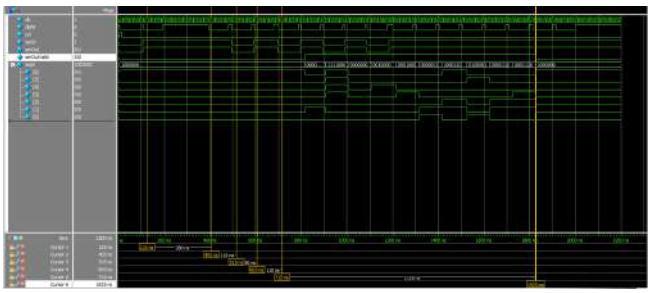

Fig. 12 Serial transmitter simulation

To see more details, the waveform is presented in two part below:

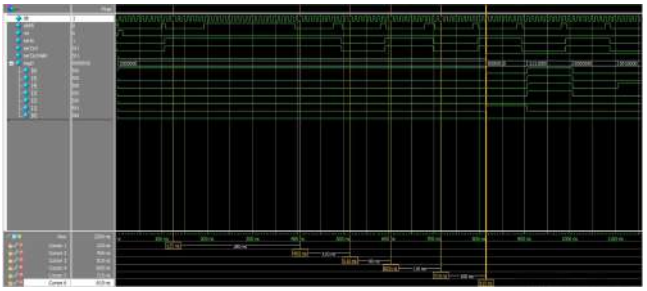

Fig. 13 Serial transmitter simulation (part 1): the circuit received 1, 1, 0, 1, 0, and 1 at cursor 1, 2, 3, 4, 5, and 6, respectively. Once the entire sequence of 110101 is detected the serOutValid signal becomes one (at cursor 6, same time as the last input is received).

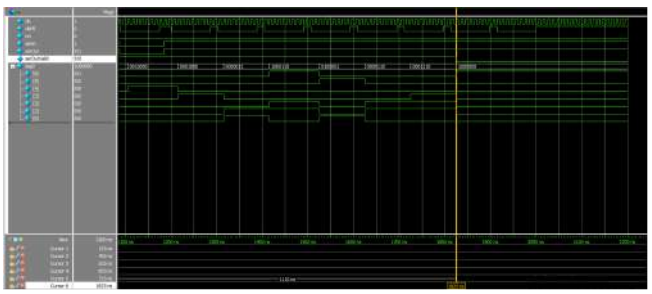

Fig. 14 Serial transmitter simulation (part2): after 10 clock cycles serOutValid signal returns to zero (at cursor 6). During this time, the counter counted from 6 to 15 and the seg0 shows its seven-segment display.

The DE1 development board was used as the hardware platform for the implementation of the design. This board provides a convenient and reliable platform for testing and verifying the functionality of the design in a real-world setting.

First, we created a Quartus project and added the top-level Verilog code to the project. After successfully compiling the design, we assigned physical pins on the Cyclone II FPGA on the DE1 board for the design. We selected the push-buttons to be used as serIn and one-pulser inputs, and determined which LEDs to use for displaying serOutValid and serOut. We then consulted the DE1 user manual to identify the position and index of each pin.

The completed pin planner window is shown in Fig. 15 which Connections are set according to table 2.

TABLE II
PIN ASSIGNMENTS

Signal Name	FPGA Signal Name	FPGA Pin No.
seg0	HEX0	-
clk	CLOCK_50	PIN_L1
clkPB	KEY[0]	PIN_R22
rst	SW[9]	PIN_L2
serIn	SW[0]	PIN_L22
serOutValid	LEDG[0]	PIN_U22
serOut	LEDR[0]	PIN_R20

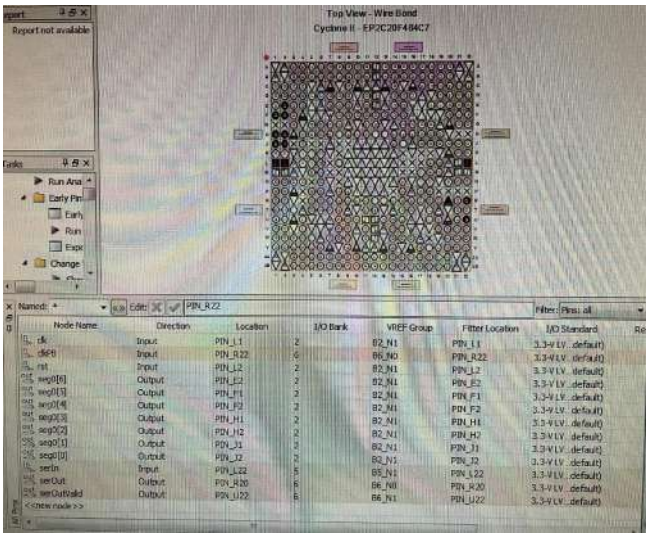

Fig. 15 Pin planner window

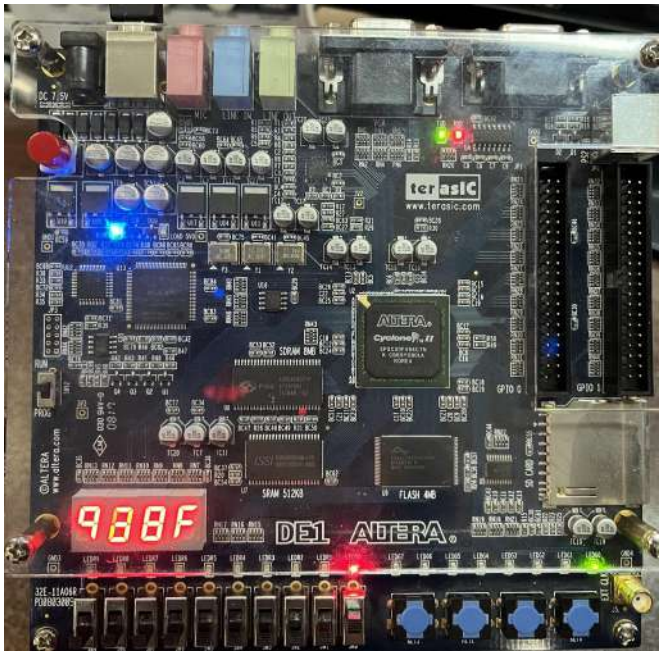

Fig. 16 A Cyclone II board with serial transmitter programmed using Quartus: Red LED(LEDR0): it shows serOut which is one at this moment as we wanted because serIn(SW0) is one. Grean LED(LEDG0): it shows serOutValid which is one at this moment, because the sequence has detected and counter output is 15 (i.e. 10 clock cycles have not passed yet). HEX0: it shows counter output.

The chip planner of each modules are Shown below which provides a visual display of device resources.

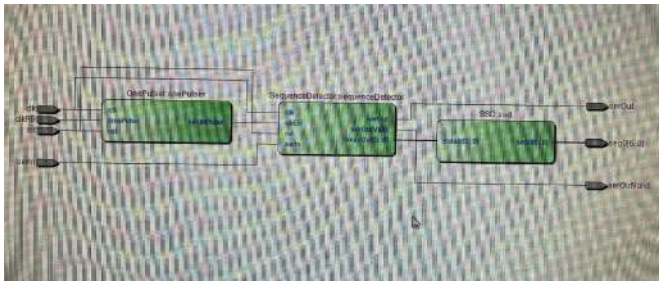

Fig. 17 Serial transmitter chip planner

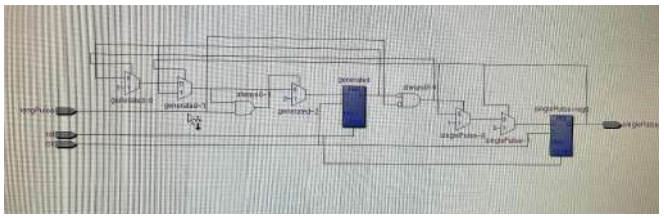

Fig. 18 One-pulser chip planner

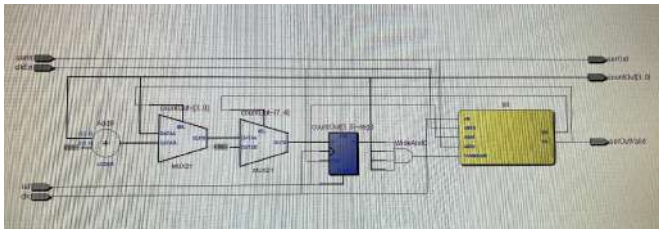

Fig. 19 Orthogonal finite state machine chip planner

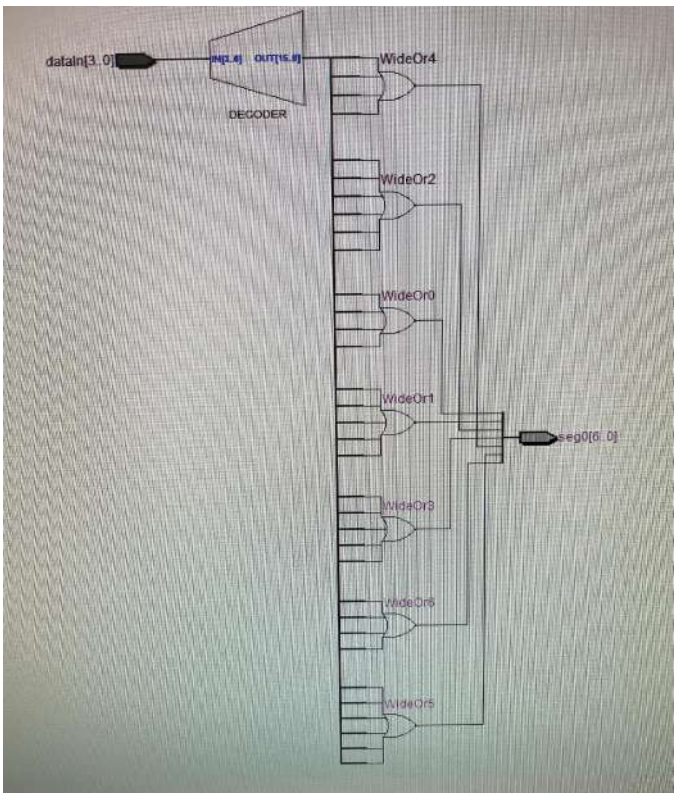

Fig. 20 Seven segment display chip planner

IV. Conclusions

Acknowledgment

This report was prepared and developed by Mehdi Jamalkhah and Mobina Mehrazar, bachelor students of Computer engineering at University of Tehran, under the supervision of professor Zain Navabi.

References