Experiment #2 - Sequential Synthesis and FPGA Device Programming

Parsa Nassery, 810199561

Shayan Kashefi, 810199585

***Abstract*— This document is a student report to experiment #2 of Digital Logic Laboratory course at ECE Department, University of Tehran. In this series of experiments, we made a transmitter that consists of three components, onepulser, OTHFSM and seven segment display. this circuit was first written in Verilog language and then it was implemented by Quartus software on an FPGA.**

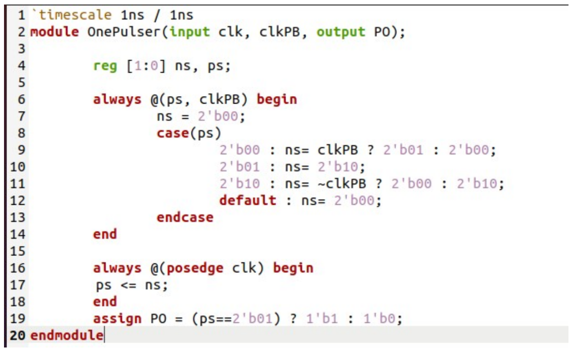
***Keywords*— FPGA programming, Serial Transmitter, Sequential Synthesis, Seven Segment, FSM.**

I.INTRODUCTION

In this experiment, a serial transmitter were writen in Verilog and then implemented on an FPGA by Quartus.

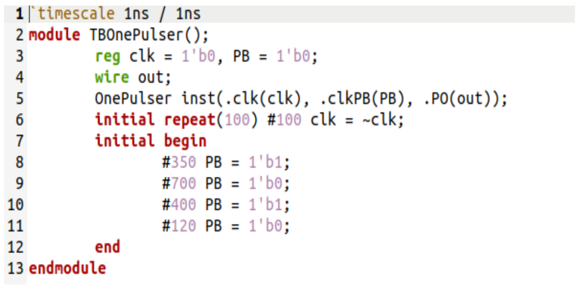
II. SERIAL TRANSMITTER

A Onepulser



```
`timescale 1ns / 1ns
module OnePulser(input clk, clkPB, output PO);

	reg [1:0] ns, ps;

	always @(ps, clkPB) begin
		ns = 2'b00;
		case(ps)
			2'b00 : ns= clkPB ? 2'b01 : 2'b00;
			2'b01 : ns= 2'b10;
			2'b10 : ns= ~clkPB ? 2'b00 : 2'b10;
			default : ns= 2'b00;
		endcase
	end

	always @(posedge clk) begin
	 ps <= ns;
	end
	assign PO = (ps==2'b01) ? 1'b1 : 1'b0;
endmodule
```

Fig. 1 Verilog description of Onepulser



```
`timescale 1ns / 1ns
module TBOnePulser();
	reg clk = 1'b0, PB = 1'b0;
	wire out;
	OnePulser inst(.clk(clk), .clkPB(PB), .PO(out));
	initial repeat(100) #100 clk = ~clk;
	initial begin
		#350 PB = 1'b1;
		#700 PB = 1'b0;
		#400 PB = 1'b1;
		#120 PB = 1'b0;
	end
endmodule
```

Fig. 2 Test bench of Onepulser

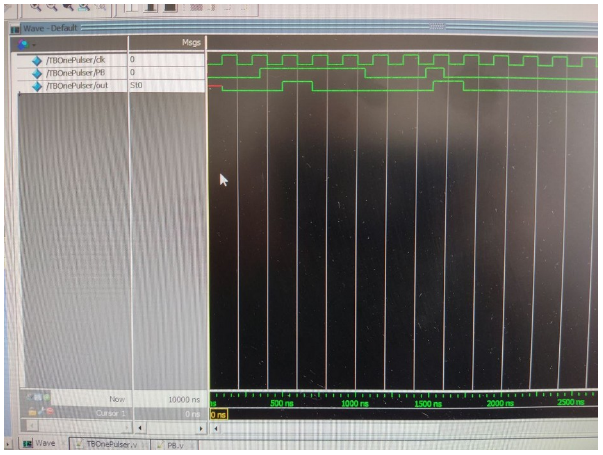


Fig. 3 Waveform of Onepulser

The Vrilog description of Onepulser is writen with design a FSM that produces the output. then a test bench was written to test this component and the output Fig. 3 was observed.

B Orthogonal Finite State Machine

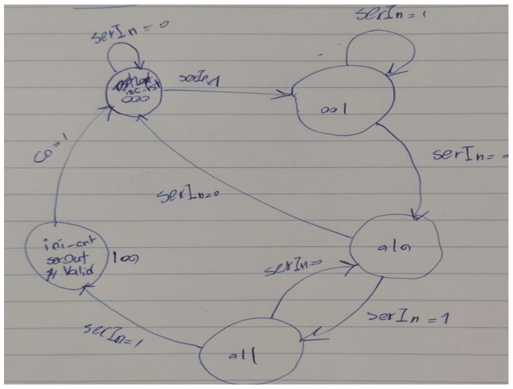
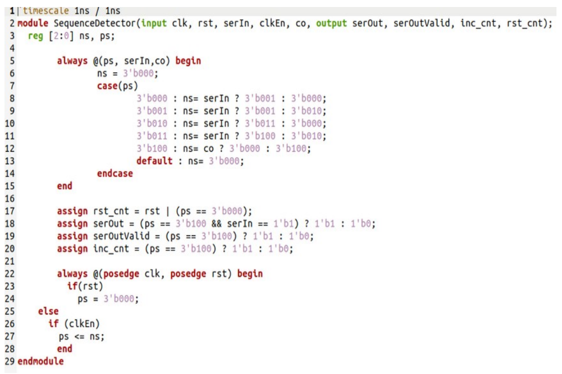
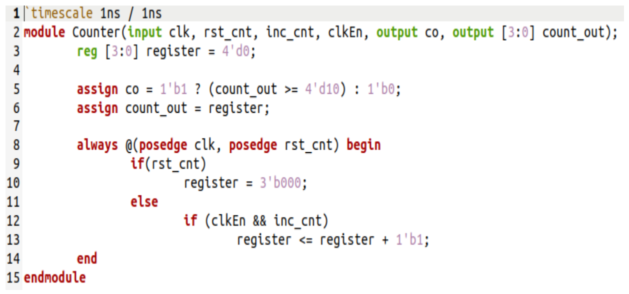


Fig. 4 State diagram of sequence detector



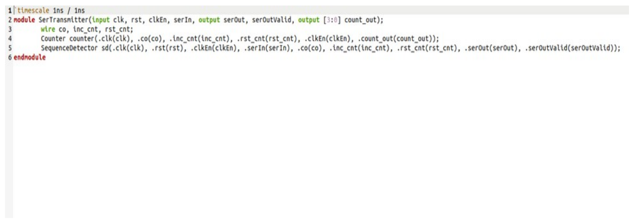
```
`timescale 1ns / 1ns
module SequenceDetector(input clk, rst, serIn, clkEn, co, output serOut, serOutValid, inc_cnt, rst_cnt);
  reg [2:0] ns, ps;

	always @(ps, serIn,co) begin
		ns = 3'b000;
		case(ps)
			3'b000 : ns= serIn ? 3'b001 : 3'b000;
			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 : ns= co ? 3'b000 : 3'b100;
			default : ns= 3'b000;
		endcase
	end

	assign rst_cnt = rst | (ps == 3'b000);
	assign serOut = (ps == 3'b100 && serIn == 1'b1) ? 1'b1 : 1'b0;
	assign serOutValid = (ps == 3'b100) ? 1'b1 : 1'b0;
	assign inc_cnt = (ps == 3'b100) ? 1'b1 : 1'b0;

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

Fig. 5 Verilog of sequence detector



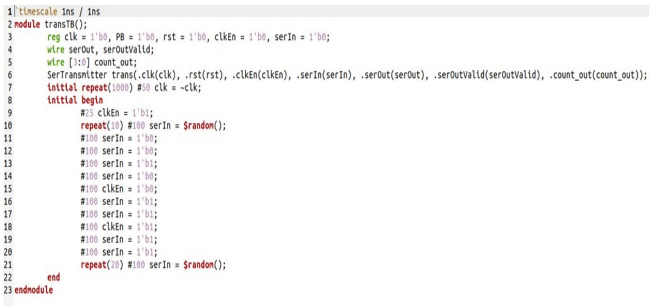
```
`timescale 1ns / 1ns
module Counter(input clk, rst_cnt, inc_cnt, clkEn, output co, output [3:0] count_out);
	reg [3:0] register = 4'd0;

	assign co = 1'b1 ? (count_out >= 4'd10) : 1'b0;
	assign count_out = register;

	always @(posedge clk, posedge rst_cnt) begin
		if(rst_cnt)
			register = 3'b000;
		else
			if (clkEn && inc_cnt)
				register <= register + 1'b1;
	end
endmodule
```

Fig. 6 Verilog of counter



```
`timescale 1ns / 1ns
module SerTransmitter(input clk, rst, clkEn, serIn, output serOut, serOutValid, output [3:0] count_out);
	wire co, inc_cnt, rst_cnt;
	Counter counter(.clk(clk), .co(co), .inc_cnt(inc_cnt), .rst_cnt(rst_cnt), .clkEn(clkEn), .count_out(count_out));
	SequenceDetector sd(.clk(clk), .rst(rst), .clkEn(clkEn), .serIn(serIn), .co(co), .inc_cnt(inc_cnt), .rst_cnt(rst_cnt), .serOut(serOut), .serOutValid(serOutValid));
endmodule
```

Fig. 7 Verilog of OTHFSM



```
`timescale 1ns / 1ns
module transTB();
	reg clk = 1'b0, PB = 1'b0, rst = 1'b0, clkEn = 1'b0, serIn = 1'b0;
	wire serOut, serOutValid;
	wire [3:0] count_out;
	SerTransmitter trans(.clk(clk), .rst(rst), .clkEn(clkEn), .serIn(serIn), .serOut(serOut), .serOutValid(serOutValid), .count_out(count_out));
	initial repeat(1000) #50 clk = ~clk;
	initial begin
		#25 clkEn = 1'b1;
		repeat(10) #100 serIn = $random();
		#100 serIn = 1'b0;
		#100 serIn = 1'b0;
		#100 serIn = 1'b1;
		#100 serIn = 1'b0;
		#100 clkEn = 1'b0;
		#100 serIn = 1'b1;
		#100 serIn = 1'b1;
		#100 clkEn = 1'b1;
		#100 serIn = 1'b1;
		#100 serIn = 1'b1;
		repeat(20) #100 serIn = $random();
	end
endmodule
```

Fig. 8 Verilog of OTHFSM test bench

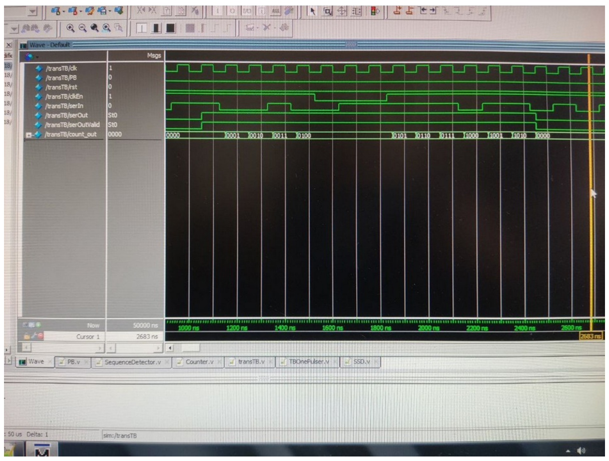
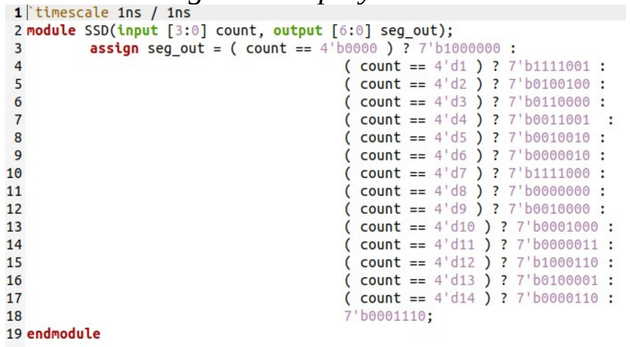


Fig. 10 Test bench of OTHFSM

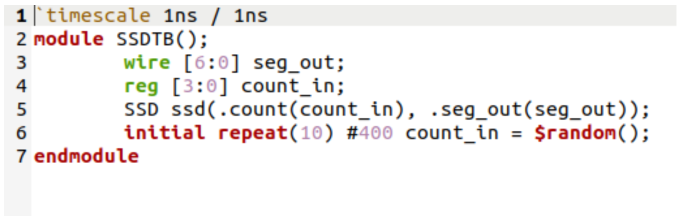
This part of the experiment consisted of two parts, one part was the sequence detector and the second part was the counter. To design the sequence detector, first, an FSM was designed that produced the desired outputs. Then the Verilog code of this FSM was written. Also, the desired counter code was written, which produced the desired output with the input signals. Finally, a test bench was written for this system and the desired outputs were observed.

C Seven Segment Display



```
`timescale 1ns / 1ns
module SSD(input [3:0] count, output [6:0] seg_out);
	assign seg_out = ( count == 4'b0000 ) ? 7'b1000000 :
				( count == 4'd1 ) ? 7'b1111001 :
				( count == 4'd2 ) ? 7'b0100100 :
				( count == 4'd3 ) ? 7'b0110000 :
				( count == 4'd4 ) ? 7'b0011001  :
				( count == 4'd5 ) ? 7'b0010010 :
				( count == 4'd6 ) ? 7'b0000010 :
				( count == 4'd7 ) ? 7'b1111000 :
				( count == 4'd8 ) ? 7'b0000000 :
				( count == 4'd9 ) ? 7'b0010000 :
				( count == 4'd10 ) ? 7'b0001000 :
				( count == 4'd11 ) ? 7'b0000011 :
				( count == 4'd12 ) ? 7'b1000110 :
				( count == 4'd13 ) ? 7'b0100001 :
				( count == 4'd14 ) ? 7'b0000110 :
				7'b0001110;
endmodule
```

Fig. 11 Verilog of SSD



```
`timescale 1ns / 1ns
module SSDTB();
	wire [6:0] seg_out;
	reg [3:0] count_in;
	SSD ssd(.count(count_in), .seg_out(seg_out));
	initial repeat(10) #400 count_in = $random();
endmodule
```

Fig. 12 Verilog of SSD test bench

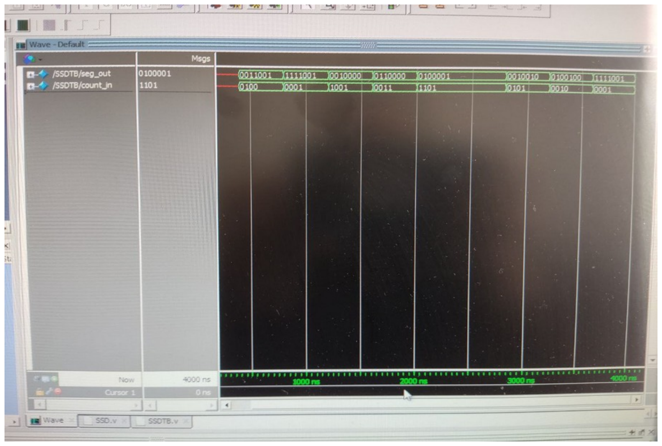
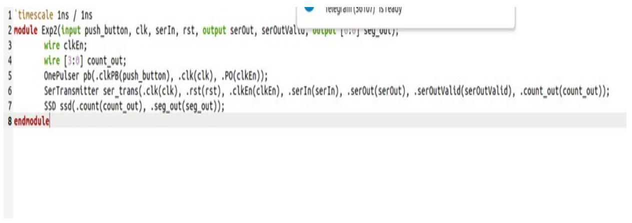


Fig. 13 Wave form of SSD

In this part, a component was designed to convert the binary output of FSM into the desired input for seven segments and a test bench is writen.

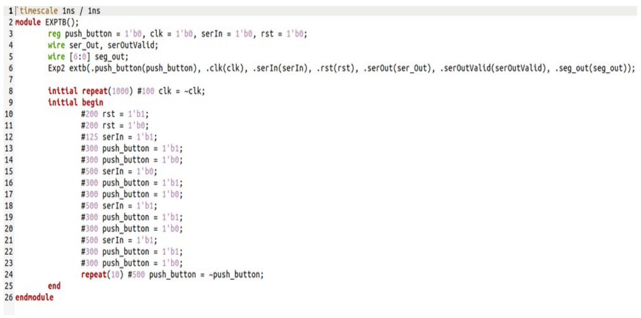
III. DESIGN SYNTHESIS AND FPGA PROGRAMMING

A. Serial Transmitter Implementation



```verilog
`timescale 1ns / 1ns
module Exp2(input push_button, clk, serIn, rst, output serOut, serOutValid, output [6:0] seg_out);
	wire clkEn;
	wire [3:0] count_out;
	OnePulser pb(.clkPB(push_button), .clk(clk), .PO(clkEn));
	SerTransmitter ser_trans(.clk(clk), .rst(rst), .clkEn(clkEn), .serIn(serIn), .serOut(serOut), .serOutValid(serOutValid), .count_out(count_out));
	SSD ssd(.count(count_out), .seg_out(seg_out));
endmodule
```

Fig. 14 Verilog of whole serial transmitter



```verilog
`timescale 1ns / 1ns
module EXPTB();
	reg push_button = 1'b0, clk = 1'b0, serIn = 1'b0, rst = 1'b0;
	wire ser_Out, serOutValid;
	wire [6:0] seg_out;
	Exp2 extb(.push_button(push_button), .clk(clk), .serIn(serIn), .rst(rst), .serOut(ser_Out), .serOutValid(serOutValid), .seg_out(seg_out));

	initial repeat(1000) #100 clk = ~clk;
	initial begin
		#200 rst = 1'b1;
		#200 rst = 1'b0;
		#125 serIn = 1'b1;
		#300 push_button = 1'b1;
		#300 push_button = 1'b0;
		#500 serIn = 1'b0;
		#300 push_button = 1'b1;
		#300 push_button = 1'b0;
		#500 serIn = 1'b1;
		#300 push_button = 1'b1;
		#300 push_button = 1'b0;
		#500 serIn = 1'b1;
		#300 push_button = 1'b1;
		#300 push_button = 1'b0;
		repeat(10) #500 push_button = ~push_button;
	end
endmodule
```

Fig. 15 Verilog of serial transmitter test bench

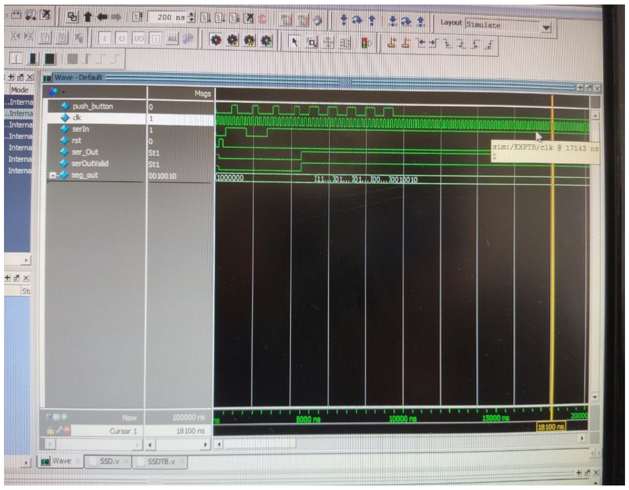


Fig. 16 Wave form of serial transmitter

First, all three components are connected to each other to make the main serial transmitter. Then a test bench was written for it. Finally, by transferring the codes to Quartos and synthesizing the circuit and assigning the pins, the final system was programmed on Ephijia and the final output was taken.

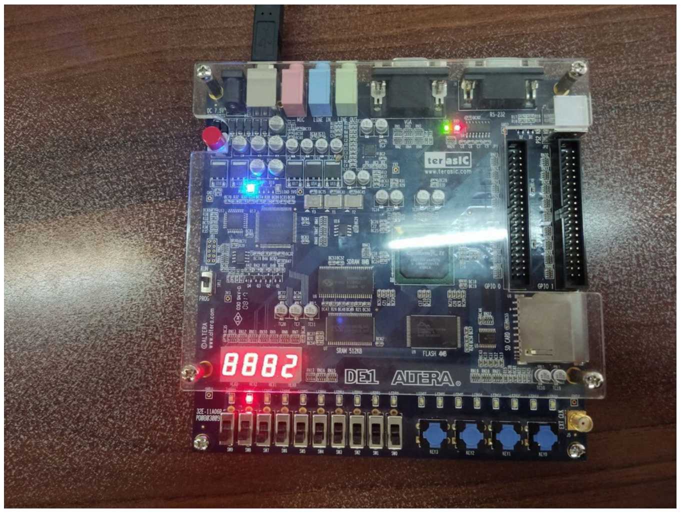


Fig. 17 Final programmed FPGA and final result