Experiment 2 - Sequential Synthesis and FPGA Device Programming

Digital Logic Design Lab - Fall 2022

Erfan Panahi
ID: 810198369

Sogol Goodarzi
ID: 810198467

SERIAL TRANSMITTER

INTRODUCTION

The first 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.

By the end of this experiment, you will learn:

- Concepts of state machines and Sequence detectors
- Huffman coding style
- Design simulation
- Synthesis
- FPGA proramming and implementation

I. ONEPULSER

Question 1. Write the Verilog description of the One-Pulser module.

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

```
module OnePulser(input clk, ClkPB, rst, output Clk_EN);
  reg [1:0] ns, ps;

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

  always @(posedge clk, posedge rst) begin
    if (rst)
      ps <= 2'b00;
    else
      ps <= ns;
  end

  assign Clk_EN = (ps == 2'b01) ? 1'b1 : 1'b0;

endmodule
```

Fig. 1. Verilog description of the One-Pulser

Question 2. Test your design in Modelsim.

For testing the designed module, a test bench has been written and it is shown in Fig. 2, and the result of testing the One-Pulser's module is shown in the waveform of Fig. 3.

```
`timescale 1ns/1ns
module OnePulser_TB;
  reg ClkPB = 0;
  reg rst;
  reg clk = 0;
  wire Clk_EN;
  OnePulser UUT (.clk(clk),.ClkPB(ClkPB),
                 .rst(rst),.Clk_EN(Clk_EN));
  initial
    begin
      #1 rst = 1'b1;
      #5 rst = 1'b0;
      #100 ClkPB = 0;
      #100 ClkPB = 1;
      #50 ClkPB = 0;
      #100 $stop;
    end

    always
    begin
      #20 clk = ~clk;
    end
  endmodule
```

Fig. 2. The test bench for testing One-Pulser module

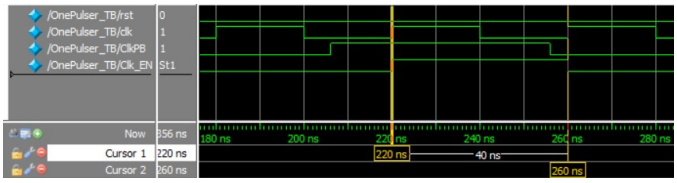

Fig. 3. Output waveform of testing the One-Pulser module

II. ORTHOGONAL FINITE STATE MACHINE

Question 1. Show the state diagram of the sequence detector.

The state diagram of the 1011 sequence detector is illustrated in Fig. 4.

Question 2. Write the top-level Verilog description of the transmitter circuit. In the top-level description, write the Verilog description of the sequence detector and the Verilog description of the counter.

The top-level Verilog description of the transmitter circuit is

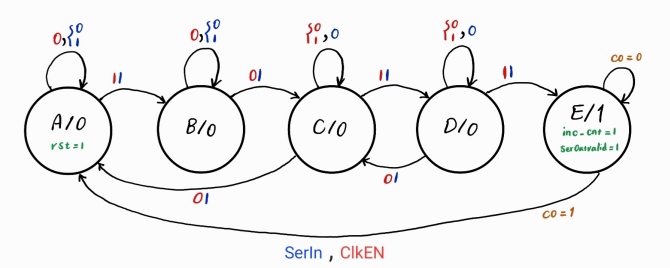

Fig. 4. State diagram of the 1011 sequence detector

shown in Fig. 5.
The Verilog description of the sequence detector is shown in Fig. 6.
The Verilog description of the counter is shown in Fig. 7.

```
`timescale 1ns/1ns
module SerialTransmitter(input clk, ClkPB, SerIn, rst,
                         output [6:0] Display,
                         output SerOut, SerOutValid);
  wire Clk_EN, Co, inc_cnt, rst_cnt;
  wire [3:0] Count_out;

  OnePulser op(clk, ClkPB, rst, Clk_EN);
  Moore1011 sd(clk, rst, Clk_EN, SerIn, Co,
               SerOut, SerOutValid, inc_cnt, rst_cnt);
  UpCnt4 cnt(clk, Clk_EN, rst_cnt, inc_cnt, Count_out, Co);
  SSD ssd(Count_out, Display);

endmodule
```

Fig. 5. The top-level Verilog description of the transmitter circuit

Question 3. Write a testbench for the serial transmitter unit in Verilog that tests the correctness of your circuit and show the results in ModelSim.
The written testbench for the serial transmitter unit that tests the correctness of circuit is shown in Fig. 8. The waveform result for testing the circuit is shown in Fig. 14 and Fig. 15.

III. Seven Segment Display

Question 1. Write the Verilog description of the seven segment display module.
The Verilog description of the seven segment display module is shown in Fig. 9.

Question 2. As an additional point, you can convert the counter output to BCD values and then display it on the seven segments using the SSD Module.
The conversion table for converting the counter output to BCD values for SSD display is shown in TABLE 1.

Design Synthesis and FPGA Programming

Serial Transmitter Implementation

First of all, we compile the Verilog descriptions in Quartus and then program the board. Secondly, we assign the input and output pins of the designed circuit to the switches and LEDs of the board. (For this aim we use the datasheet of the board for pin assignment.) The assigned pins are shown in TABLE

```
module Moore1011(input clk, rst, Clk_EN, SerIn, Co,
                 output SerOut,
                 output reg SerOutValid, inc_cnt, rst_cnt);
  reg [2:0] ns, ps;
  always @(ps, SerIn, Clk_EN, Co) begin
    ns = 3'b000;
    case (ps)
      3'b000: ns = (Clk_EN & SerIn) ? 3'b001 : 3'b000;
      3'b001: ns = (Clk_EN & ~SerIn) ? 3'b010 : 3'b001;
      3'b010: begin
        if (Clk_EN == 1)
          ns = SerIn ? 3'b011 : 3'b000;
        else
          ns = 3'b010;
      end
      3'b011: begin
        if (Clk_EN == 1)
          ns = SerIn ? 3'b100 : 3'b010;
        else
          ns = 3'b011;
      end
      3'b100: ns = Co ? 3'b000 : 3'b100;
      default: ns = 3'b000;
    endcase
  end

  always @(ps) begin
    {SerOutValid, inc_cnt, rst_cnt} = 3'b000;
    case (ps)
      3'b000: rst_cnt = 1'b1;
      3'b100: {SerOutValid, inc_cnt} = 2'b11;
      default: {SerOutValid, inc_cnt, rst_cnt} = 3'b000;
    endcase
  end
  assign SerOut = SerOutValid ? SerIn : 1'bz;

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

Fig. 6. The Verilog description of the sequence detector

```
`timescale 1ns/1ns

module UpCnt4(input clk, Clk_EN, rst_cnt, inc_cnt,
              output reg[3:0]Count_out, output Co);
  always @(posedge clk, posedge rst_cnt) begin
    if (rst_cnt)
      Count_out <= 4'd5;

   else if (inc_cnt)
      Count_out <= Clk_EN ? (Count_out+1) : Count_out;
  end
  assign Co = (inc_cnt) ? &{Count_out,Clk_EN} : 1'b0;

endmodule
```

Fig. 7. The top-level Verilog description of the counter

2. Also for the outputs ”display” in seven segment module we assign HEX0[0:3] to them. For testing the circuit, the switches should be altered to 0 and 1 values in a way that the sequence detector detects a 1011 and then after that in 10 consecutive clocks, the counter can count from 5 to F (15) and it can be seen through seven segments.

The result of implementing on FPGA board is as the same as the output reached by simulation (TestBench). The observations of board functionality are shown in Fig. 10. to 13.

```
`timescale 1ns/1ns
module SerialTransmitter_TB;
    reg rst, clk = 0, clkPB, SerIn;
    wire [6:0] Display;
    wire SerOut, SerOutValid;

    SerialTransmitter ST(clk, clkPB, SerIn,
                         rst, Display, SerOut,
                         SerOutValid);

    always begin
        #10 clk=~clk;
    end

    initial begin
        #1 rst = 1;
        #1 rst = 0;
        #1 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #11 clkPB = 0;
        #20 SerIn = 0;
        #1 clkPB = 1;
        #10 SerIn = 1;
        #52 SerIn = 0;
        #1 clkPB = 1;
        #21 clkPB = 0;
        #21 clkPB = 1;
        #10 SerIn = 1;
        #11 clkPB = 0;
        #42 clkPB = 1;
        #41 clkPB = 0;
        #41 clkPB = 1;
        #41 clkPB = 0;
        #41 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 0;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 0;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 0;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
        #21 clkPB = 1;
        #10 SerIn = 0;
        #10 SerIn = 1;
```

Fig. 8. Testbench for the serial transmitter unit

TABLE I
THE CONVERSION TABLE FOR CONVERTING THE COUNTER OUTPUT TO BCD VALUES

Binary Values	***Seven Segment Display***
0000	1000000
0001	1111001
0010	0100100
0011	0110000
0100	0011001
0101	0010010
0110	0000010
0111	1111000
1000	0000000
1001	0010000
1010	0001000
1011	0000011
1100	1000110
1101	0100001
1110	0000110
1111	0001110

TABLE II
PIN ASSIGNMENT OF THE BOARD

Pin on board	***Pin in module***
$SW[0]$	$SerIn$
$SW[1]$	$ClkPB$
$SW[2]$	rst
$LEDR[8]$	$SerOut$
$LEDR[1]$	$SerOutValid$

```
`timescale 1ns/1ns
module SSD(input [3:0]Count_out,
           output reg[6:0]Display);

  always @(Count_out) begin
    case(Count_out)
      4'b0000: Display = 7'b1000000;
      4'b0001: Display = 7'b1111001;
      4'b0010: Display = 7'b0100100;
      4'b0011: Display = 7'b0110000;
      4'b0100: Display = 7'b0011001;
      4'b0101: Display = 7'b0010010;
      4'b0110: Display = 7'b0000010;
      4'b0111: Display = 7'b1111000;
      4'b1000: Display = 7'b0000000;
      4'b1001: Display = 7'b0010000;
      4'b1010: Display = 7'b0001000;
      4'b1011: Display = 7'b0000011;
      4'b1100: Display = 7'b1000110;
      4'b1101: Display = 7'b0100001;
      4'b1110: Display = 7'b0000110;
      4'b1111: Display = 7'b0001110;
    endcase
  end

endmodule
```

Fig. 9. The Verilog description of the seven segment display module

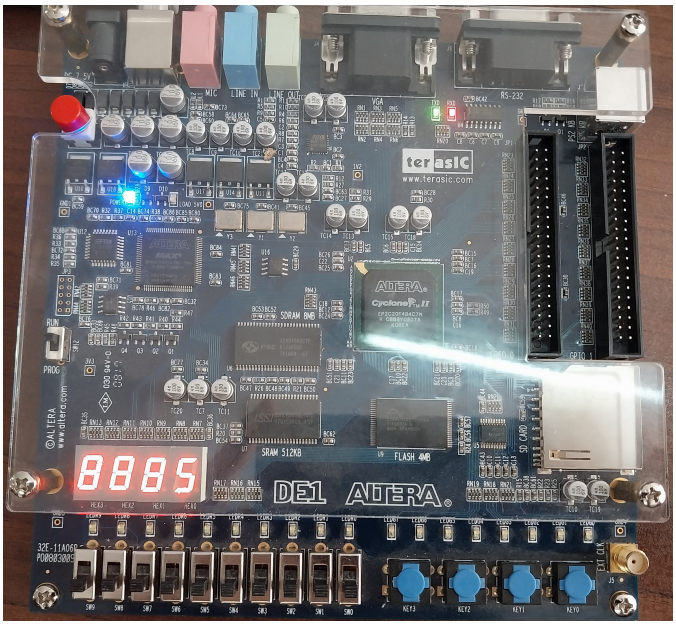

Fig. 10. The 1011 sequence is detected here!

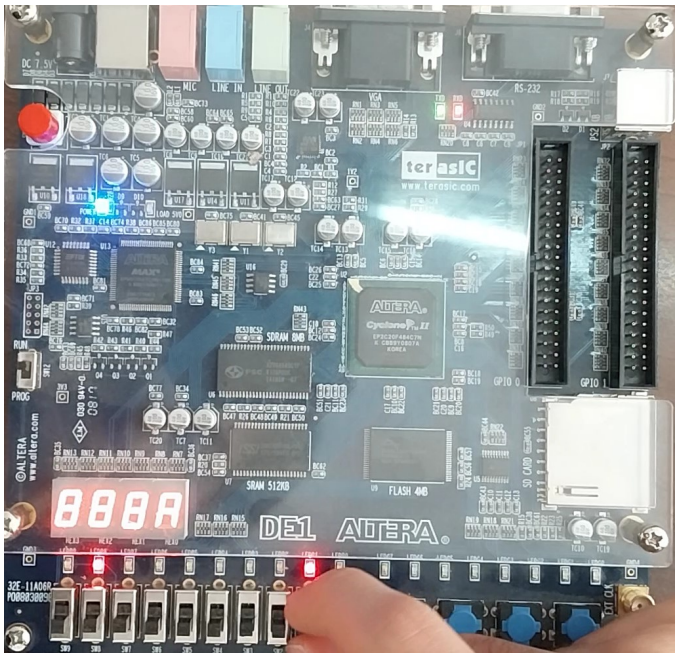

Fig. 12. The 1011 sequence is detected here!

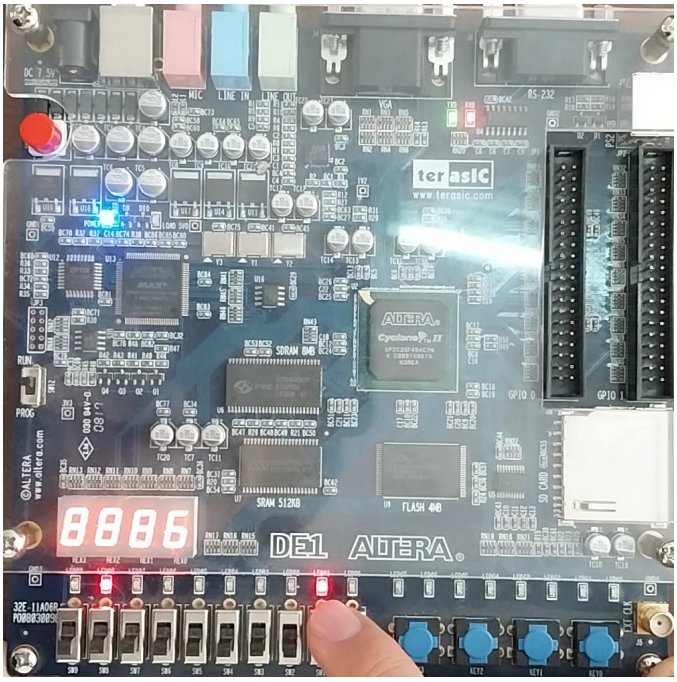

Fig. 11. The counter starts counting here!

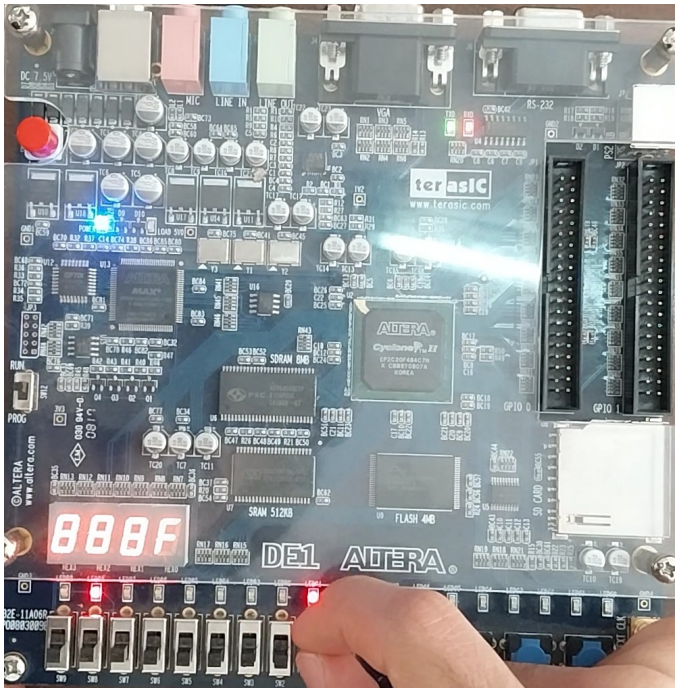

Fig. 13. The counter starts counting here!

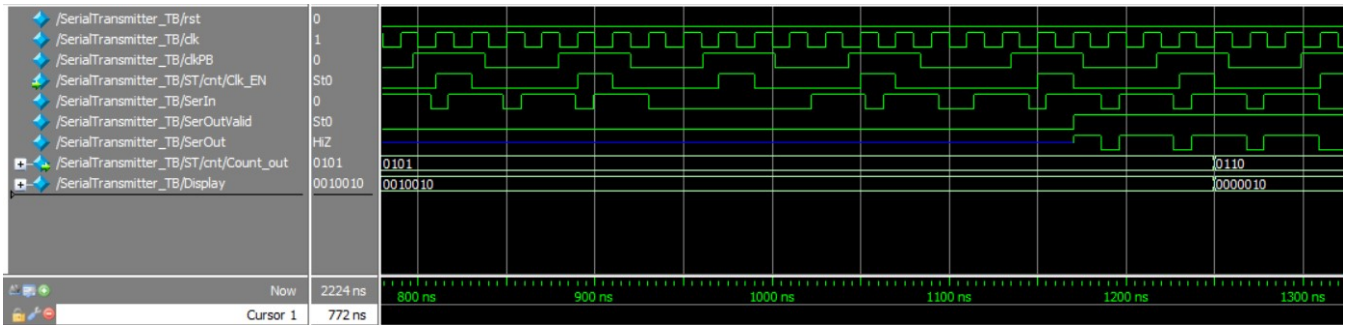

Fig. 14. The 1011 sequence is detected here!

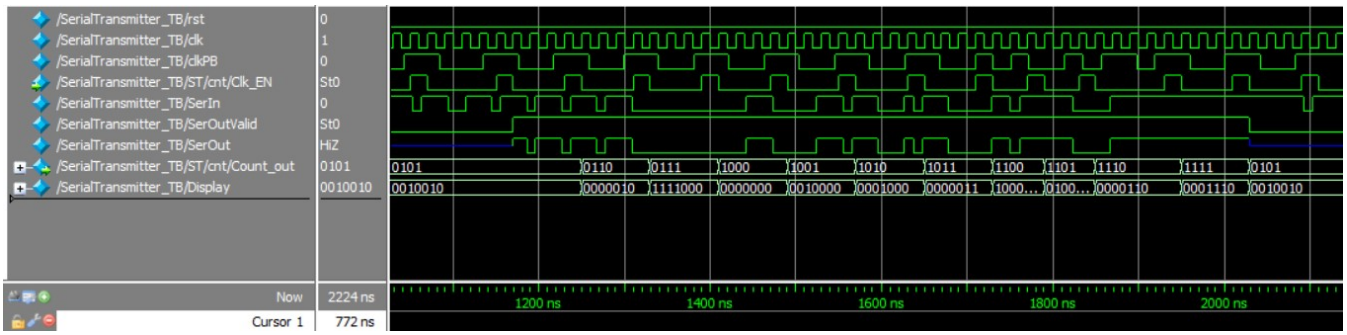

Fig. 15. The counter starts counting here!