Experiment 2 – Sequential Synthesis and FPGA Programming

Amirhosein Yavarikhoo 810199514

***Abstract*— In this experiment we use FPGA boards for the first time. The circuit we design is used to detect a sequence; When detected, it puts the input on output for 10 clocks.**

Keywords, FPGA board, sequence detector, counter, one pulse machine

I. Introduction

FPGA boards are used in many different areas in circuit implementation. In this experiment, we program a FPGA board to manufacture our circuit design.

II. Serial Transmitter

This serial transmitter detects a 110101 sequence. After detecting, valid flag becomes 1 and for the next 10 cycles serial input will be transmitted on serial output. When transmitting is finished, the circuit looks for another start sequence.

III. One Pulser

FPGA boards work on a high frequency; Therefore, the internal clock in boards is very fast and we can't see the results we're expecting. In order to solve this problem, we use a one pulser device output as a clock enable input for every RTL component used in this circuit. One pulser detects a 010 sequence and creates a single one pulse on the output. This way, we can create a manual clock to see each state and output easily.

In Fig.1 state diagram for One Pulser is shown.

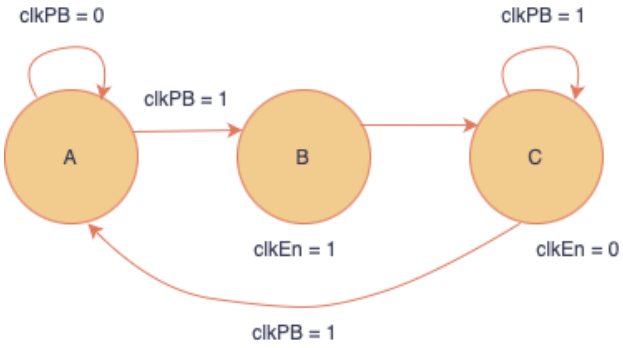

Fig. 1 One Pulser State Diagram

In Fig.2 Verilog code for One Pulser is shown:

```
Ln#
1  module one_pulser( input clk, rst, clkPB, output reg clkEn );
2    //  parameter [1:0] A=0, B=1, C=2;
3      reg [1:0] ps, ns;
4
5      always @(posedge clk, posedge rst) begin
6
7    ns = 2'b0;
8    case (ps)
9     2'b0 :       ns = clkPB ? 2'b01 : 2'b0;
10    2'b01 : begin ns = 2'b10; clkEn = 1'b1; end
11    2'b10 : begin ns =  clkPB ? 2'b10 : 2'b0; clkEn = 1'b0; end
12    default: ns = 2'b0;
13    endcase
14      end
15
16     always @(posedge clk, posedge rst) begin
17   if(rst)
18    ps <= 2'b0 ;
19   else
20    ps <= ns;
21      end
22
23
24 endmodule
25
```

Fig. 2 One Pulser Verilog Code

Now, we create a testbench for this device to show its functionality.

```
Ln#
1   `timescale 1ns/1ns
2   module one_pulserTB();
3    reg clkPB = 0,rst = 0,clk =0;
4    one_pulser CUT1( clk,rst,clkPB,clkEn);
5    always #5 clk = ~clk;
6    initial begin
7    #12 rst = 1'b1;
8    #5 rst = 1'b0;
9    #19 clkPB = 1'b0;
10   #22 clkPB =1'b1;
11   #91 clkPB = 1'b0;
12   #19 clkPB = 1'b0;
13   #22 clkPB =1'b1;
14   #91 clkPB = 1'b0;
15   //#20 rst = 1'b1;
16   #40 $stop;
17    end
18  endmodule
19
```

Fig. 3 One Pulser Testbench

Testbench waveform is shown below:

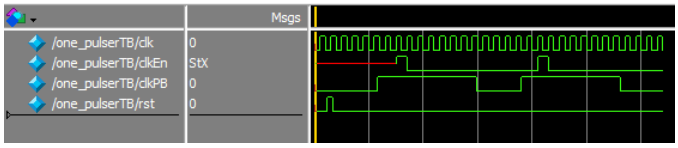

Fig. 4 One Pulser Waveform

IV. Orthogonal Finite State Machine

In this experiment we detect a 110101 sequence. To implement this design, we use a Finite State Machine (FSM) in a Moore style. The diagram representing states of this machine is shown below:

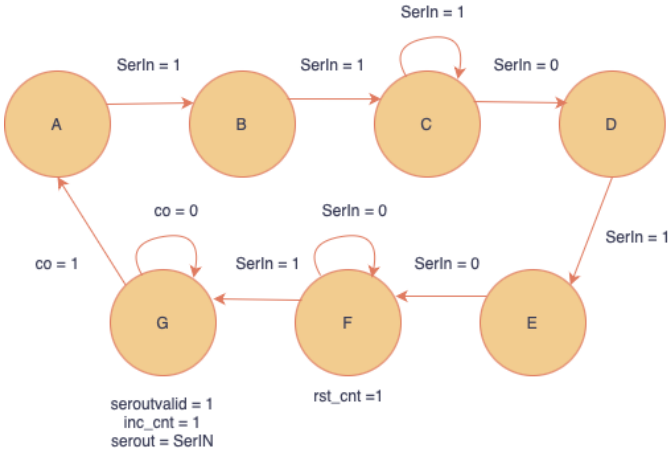

Fig. 5 Sequence Detector State Machine

According to this diagram, we program this device in Verilog.

```
Ln#
1  module seqdtc(input clk_EN,SerIn,clk,rst,co,output reg seroutvalid,serout,inc_cnt,rst_cnt);
2  reg [2:0] pstate,nstate;
3  parameter [2:0] A=0,B=1,C=2,D=3,E=4,F=5,G=6;
4  always@(pstate,co) begin
5  seroutvalid=1'b0;
6  inc_cnt=1'b0;
7  rst_cnt=1'b0;
8  //pstate=3'd0;
9  case (pstate)
10 A: nstate=SerIn?B:A;
11 B: nstate=SerIn?C:A;
12 C: nstate=SerIn?C:D;
13 D: nstate=SerIn?E:A;
14 E: nstate=SerIn?C:F;
15 F: begin nstate=SerIn?G:A; rst_cnt=1'b1; end
16 G: begin
17         seroutvalid=1'b1;
18         inc_cnt=1'b1;
19         serout=SerIn;
20         if (co) nstate=A;
21         else nstate=G;
22 end
23 default : nstate=A;
24 endcase
25 end
26 always @(posedge clk) begin
27 if (rst==1'b1) begin
28 pstate=3'd0;
29 end
30 if (clk_EN==1'b1) pstate=nstate;
31 end
32 endmodule
```

Fig. 6 Sequence Detector Verilog Code

Now, we use a counter to count to 10. In order to do this, we set the initial value of the counter at 6 and whenever carry out (co) becomes 1, we've counted to 10.

Verilog code of this counter is shown below:

```
Ln#
1  module upcounter(input clk_EN,clk,inc_cnt,rst_cnt, output reg [3:0] count_out, output reg co);
2  always @(posedge clk) begin
3          if (rst_cnt) begin
4                  count_out=4'd6;
5                  co=1'b0;
6                  end
7          if (inc_cnt&clk_EN) begin
8                  count_out=count_out+1;
9                  co=(count_out==4'd15)?1:0;
10         end
11 end
12 endmodule
```

Fig. 7 Counter Verilog Code

The last component is a hex display to display the counter. Verilog code of this display is shown below:

```
Ln#
1  module hexdisplay(input [3:0] number, output reg [6:0] segments);
2  always @(number) begin
3          case (number)
4                          4'd0:segments=7'b1000000;
5
6                          4'd1:segments=7'b1111001;
7
8                          4'd2:segments=7'b0100100;
9
10                         4'd3:segments=7'b0110000;
11
12                         4'd4:segments=7'b0011001;
13
14                         4'd5:segments=7'b0010010;
15
16                         4'd6:segments=7'b0000010;
17
18                         4'd7:segments=7'b1111000;
19
20                         4'd8:segments=7'b0000000;
21
22                         4'd9:segments=7'b0010000;
23
24                         default:segments=7'b100_0000;
25
26         endcase
27
28 end
29 endmodule
```

Fig. 8 Hex Display Verilog Code

Now that we have all the components, we connect them together to create a serial transmitter and test it to make sure of its functionality.

```
Ln#
1  module serialtransmitter (input clk,rst,clk_PB,SerIn, output serout,seroutvalid, output [3:0] count_out, output [6:0] segments);
2          wire inc_cnt,co,clk_EN;
3          upcounter CNT (clk_EN,clk,inc_cnt,rst,count_out,co);
4          seqdtc DTC (clk_EN,SerIn,clk,rst,co,seroutvalid,serout,inc_cnt,rst_cnt);
5          one_pulser ONEP(clk, rst, clk_PB,clk_EN );
6          hexdisplay disp (count_out,segments);
7  endmodule
8
```

Fig. 9 Serial Transmitter Module

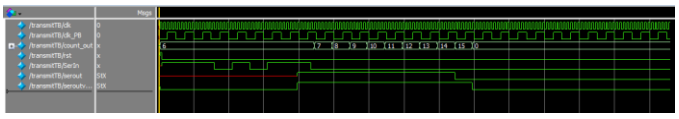

Fig. 10 Serial Transmitter Waveform

As we can see, this circuit works as intended.

V. IMPLEMENTATION

We implement this project on FPGA board. Table.1 shows the component assignment for this board:

TABLE I

FPGA COMPONENT ASSIGNMENTS

Signal	Board implementation
serIN	Switch 0
clkPB	Switch 1
Rst	Switch 2
Seroutvalid	LED Red 0
serout	LED Green 0
Hex display	FPGA monitor

Images of the board is shown below:

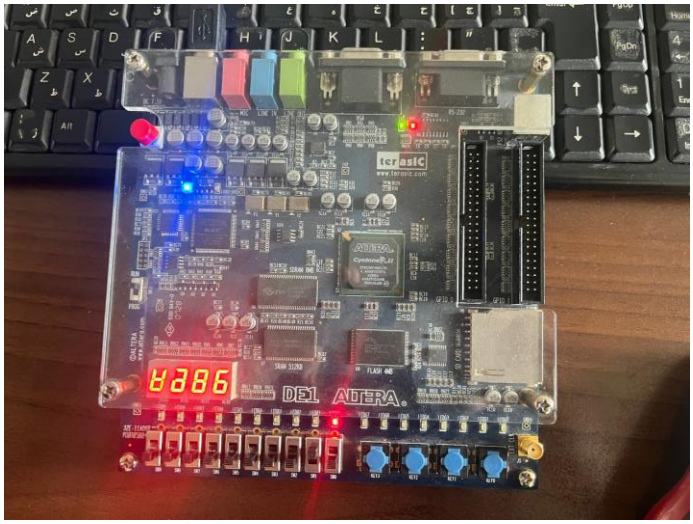

Fig. 11 Implementation of FPGA

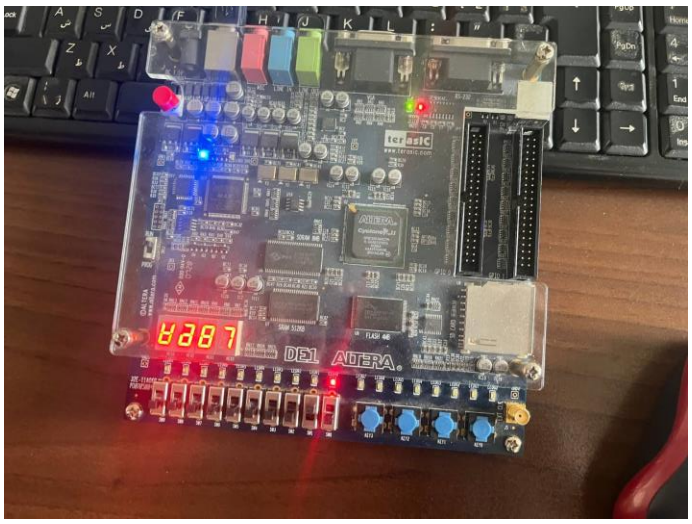

Fig. 12 Implementation of FPGA

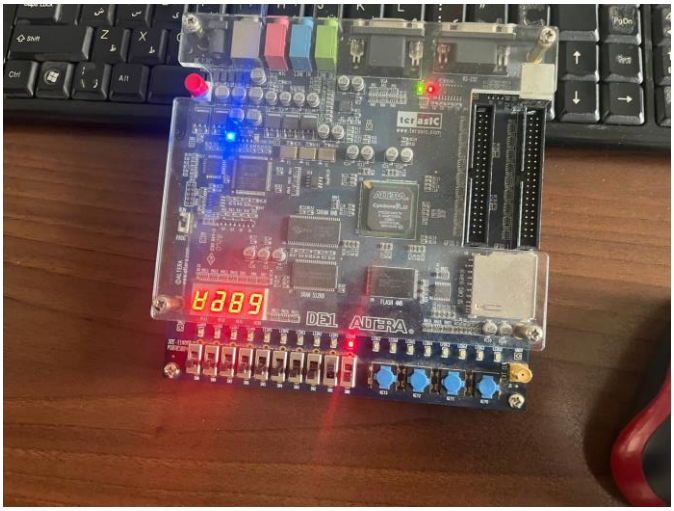

Fig.13 Implementation of FPGA