Experiment 2 – Sequential Synthesis and FPGA Programming

Matin Bazrafshan, 810100093

Mohammad Amin Yousefi, 810100236

Abstract— **This document is a report for experiment 2 of Digital Design Laboratory at ECE department at University of Tehran. The purpose of this experiment is to design a sequence detector and synthesize on an FPGA .**

***Keywords*—DE1 FPGA Sequence Detector, OnePulser, Orthogonal Finite State Machine, Quartus, Serial Transmitter**

I. Introduction

In this experiment we will design a sequence detector with a finite state machine with Verilog and implement it on an FPGA.

II. Serial Transmitter

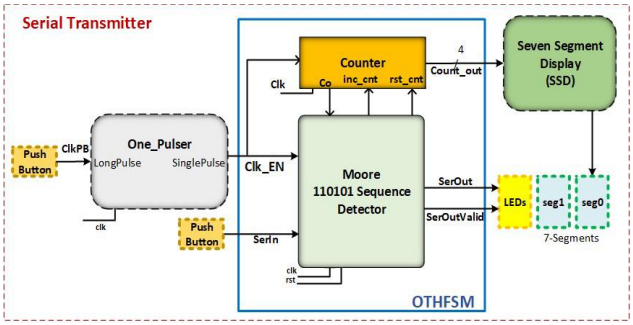


Fig. 1 schematic view of serial transmitter

A. One Pulser

One pulser is module which provides a clock-enable signal as output which is active in only one clock, so we can use it when we want to use a button as a clock, in a single push button, we have many clock due to its high frequency so we need a one-pulser. Clock-enable is used when we want to implement our circuit on an FPGA. On any push button, it creates only a single clock.

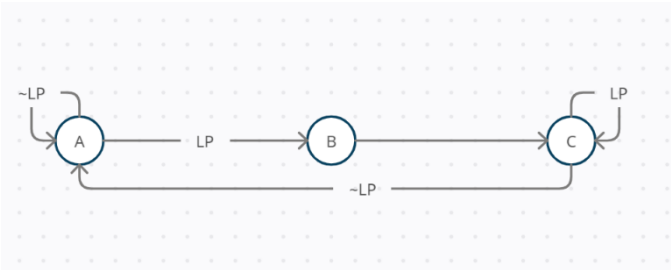


Fig. 2 One-Pulser States diagram

```verilog
module one_pulser (
    input clkPB,clk,rst,
    output reg SP
);
    parameter A = 0, B = 1, C = 2;
    reg [1:0] pstate,nstate;
    initial pstate = A;
    always @(pstate, clkPB) begin
        SP = 0;
        case (pstate)
            A: nstate = clkPB ? B : A;
            B: begin
                    nstate = C;
                    SP = 1;
                end
            C: nstate = clkPB ? C : A;
            default: nstate = A;
        endcase
    end
    always @(posedge clk, posedge rst) begin
        if(rst)
            pstate <= A;
        else
            pstate <= nstate;
    end
endmodule
```

Fig. 3 One-Pulser Verilog code

```verilog
module one_pulse_TB;
    wire out;
    reg LP,clk = 0;
    one_pulser op(LP,clk,out);
    always begin
        #5 clk = ~clk;
    end
    initial begin
        #7 LP = 1;
        #100 LP = 0;
    end

endmodule
```

Fig. 4 One-Pulser Test Bench Verilog Code

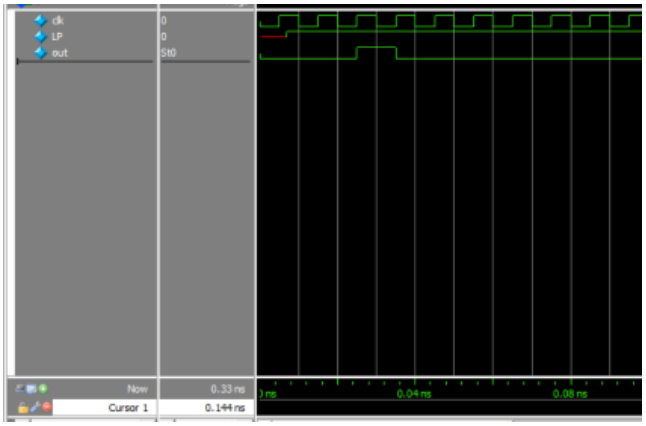


Fig. 5 One-Pulser Waveform

B. Orthogonal Finite State Machine

Orthogonal FSM consists a Moore state machine and a counter. The sequence detector detects 110101 on its SerIn input and after it receives it, SerOutValid becomes 1, After that, the counter will be enabled and will count for the next 10 consecutive clock cycles. During all these time the serOutvalid remains one and SerOut will be displayed on output.

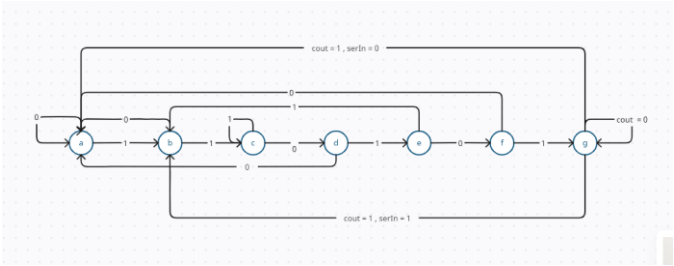


Fig. 6 sequence detector state diagram

```verilog
module sequence_detector (
    input clk,rst,clkEN,cout,SerIn,
    output SerOut,
    output reg SerOutValid,inc_cnt,rst_cnt
);
    parameter A = 0, B = 1, C = 2, D = 3, E = 4, F = 5, G = 6;
    reg [2:0] pstate,nstate;
    initial pstate = A;
    always @(pstate, SerIn, cout) begin
        {inc_cnt, rst_cnt, SerOutValid} = 3'b000;
    $display("%d",pstate);
        case(pstate)
            A: nstate = SerIn ? B : A;
            B: nstate = SerIn ? C : A;
            C: nstate = SerIn ? C : D;
            D: nstate = SerIn ? E : A;
            E: nstate = SerIn ? B : F;
            F: begin
                rst_cnt = 1;
                nstate = SerIn ? G : A;
            end
            G: begin
                inc_cnt = 1;
                SerOutValid = 1;
                if(cout == 1 && SerIn == 1)
                    nstate = B;
                else if(cout == 1 && SerIn == 0)
                    nstate = A;
                else if(cout == 0)
                    nstate = G;
            end
            default nstate = A;
        endcase
    end


    assign SerOut = SerIn;

    always @(posedge clk, posedge rst) begin
        if(rst)
            pstate <= A;
        else if(clkEN)
            pstate <= nstate;
    end
endmodule
```

Fig. 7 sequence detector Verilog code

```verilog
module four_bit_counter(
    input clk, rst_cnt, inc_cnt, clkEN,
    output reg [3:0] po, output cout
);

 always@ (posedge clk) begin
    if(rst_cnt)
        po <= 4'b0000;
    else if(inc_cnt && clkEN)
        po <= po + 1;
 end

 assign cout = (po == 4'b1001) ? 1 : 0;

endmodule
```

Fig. 8 4-bit counter Verilog code

```
module seq_TB;
    reg clkEN=0,SerIn,clk = 0,rst,cout = 0;
    wire SerOut,SerOutValid,inc_cnt,rst_cnt;

    sequence_detector sd(clk,rst,clkEN,cout,SerIn,SerOut
                         ,SerOutValid,inc_cnt,rst_cnt);

    always begin
        #5 clk = ~clk;

    end

    initial begin
        #23 SerIn = 1;
        #23 SerIn = 0;

        #23 clkEN = 1; SerIn = 0; #5 clkEN = 0;


        #23 clkEN = 1; SerIn = 1; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 1; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 0; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 1; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 0; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 1; #5 clkEN = 0;

        #23 clkEN = 1; SerIn = 1; #5 clkEN = 0;
        #23 clkEN = 1; SerIn = 0; #5 clkEN = 0;

        #23 clkEN = 1; cout = 1; #5 clkEN = 0;
        #23 SerIn = 1;
    end
endmodule
```

Fig. 9 sequence detector test bench Verilog code

```
module hex_display (
    input [3:0] data_in,
    output reg [6:0] hex
);
    always@(data_in) begin
        case (data_in)
            4'b0000: hex <= 7'b1000000;
            4'b0001: hex <= 7'b1111001;
            4'b0010: hex <= 7'b0100100;
            4'b0011: hex <= 7'b0110000;
            4'b0100: hex <= 7'b0011001;
            4'b0101: hex <= 7'b0010110;
            4'b0110: hex <= 7'b0000010;
            4'b0111: hex <= 7'b1111000;
            4'b1000: hex <= 7'b0000000;
            4'b1001: hex <= 7'b0010000;
            4'b1010: hex <= 7'b0001000;
            4'b1011: hex <= 7'b0000011;
            4'b1100: hex <= 7'b1000110;
            4'b1101: hex <= 7'b0100001;
            4'b1110: hex <= 7'b0000110;
            4'b1111: hex <= 7'b0001110;
            default: hex <= 7'b1000000;
        endcase
    end
endmodule
```

Fig. 10 sequence detector test bench Verilog code

C. Seven Segment Display

This module is used to display 4-bit counter output as a HEX number on FPGA.

D. Connecting All modules and Writing a Test Bench

Writing a test bench to test all modules together.

```
module data_path(input clk, rst, clkPB, SerIn, output [7:0] out, output SerOutValid, SerOut);

wire clkEN, rst_counter, inc_cnt, cout;
wire [3:0] cnt_out;

four_bit_counter FourBitCounter(clk, rst_counter, inc_cnt, clkEN, cnt_out, cout);

one_pulser OnePulser(clkPB, clk, rst, clkEN);

hex_display HexDisplay(cnt_out, out);

sequence_detector SequenceDetector (clk, rst, clkEN, cout, SerIn, SerOut, SerOutValid, inc_cnt, rst_counter);

endmodule
```

Fig. 11 All modules connected Verilog code

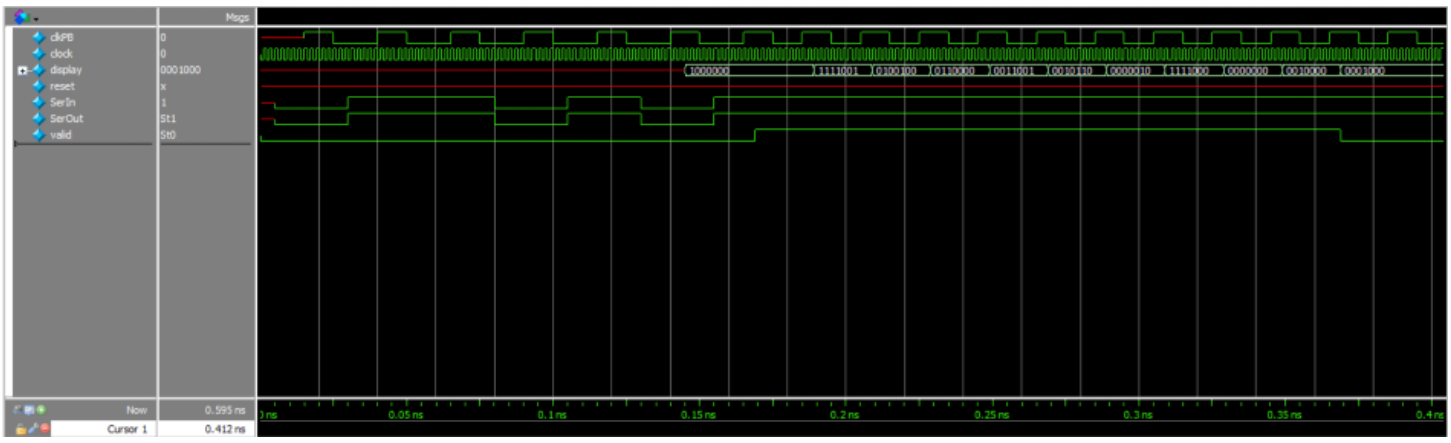


Fig. 12 All modules connected Waveform

```
module totall_TB;
    reg clock = 0,reset,clkPB,SerIn;
    wire [6:0] display;
    wire valid, SerOut;
    data_path dp(clock,reset,clkPB,SerIn,display,valid,SerOut);

    always begin
        #1 clock = ~clock;
    end

    initial begin
        #5 SerIn = 0;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 1;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 1;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 0;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 1;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 0;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #5 SerIn = 1;
        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

        #10 clkPB = 1;
        #10 clkPB = 0;

    #100 $stop;
    end
endmodule
```

Fig. 13 All modules connected test bench Verilog code

III. Synthesis and FPGA Implementation

In this part we should synthesize design on Quartus and implement it on DE1 FPGA.

Fig. 14 DE1 FPGA board counted up to 3 with SerOutValid and showing SerIn on SerOut

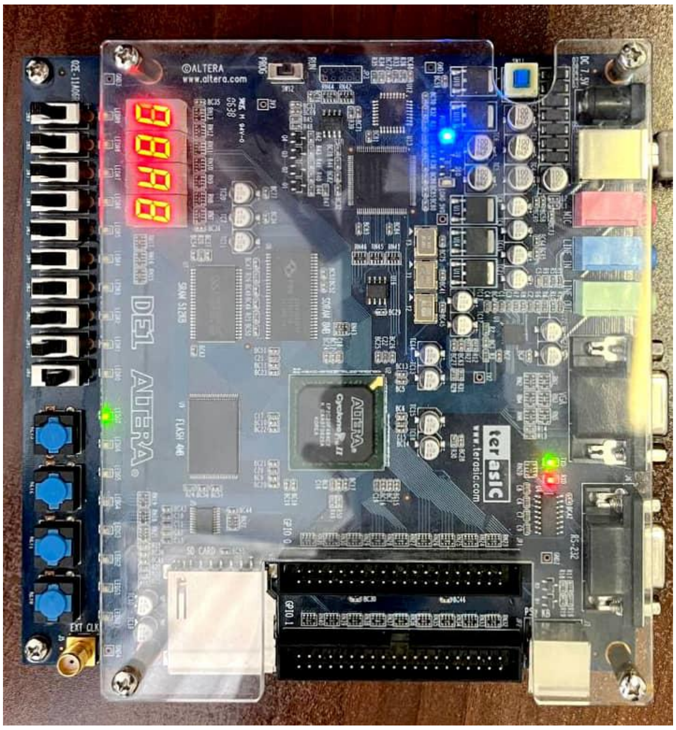
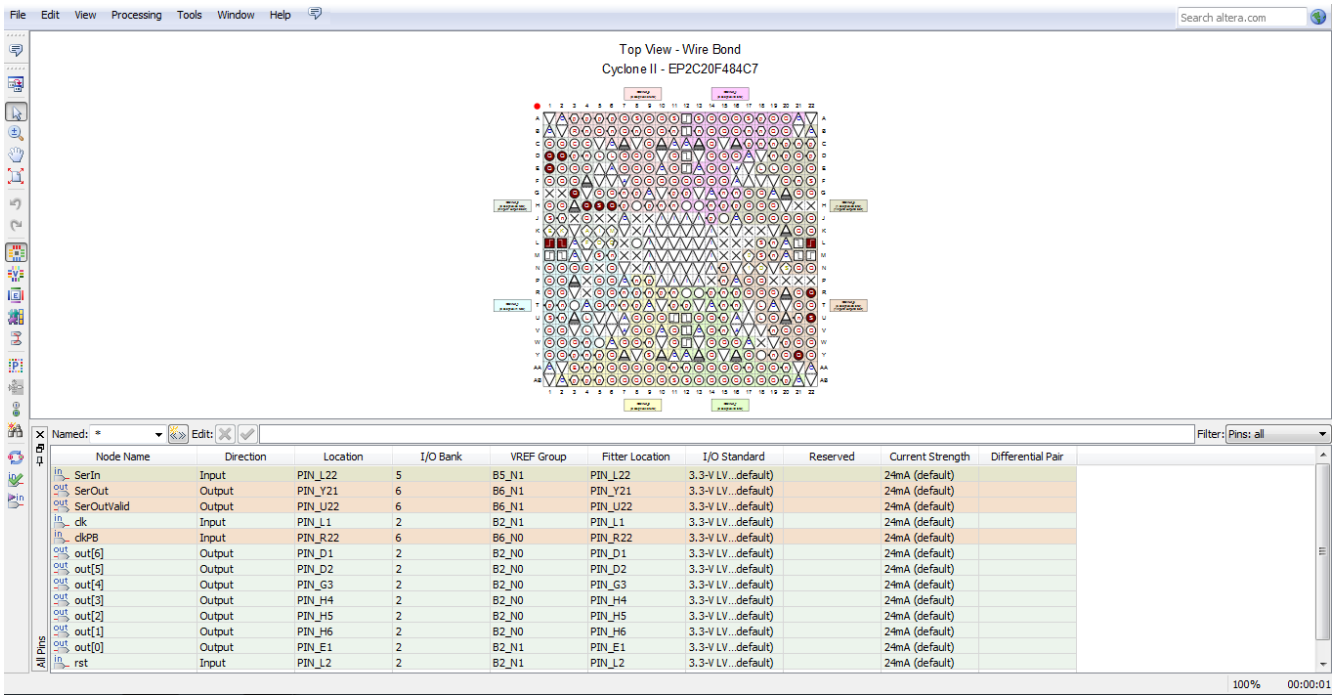


Fig. 15 DE1 FPGA board counted up to 10 with SerOutValid and showing SerIn on SerOut

The leftmost LED is for SerOutValid and the rightmost is SerOut which displays SerIn directly.
The second Seven Segment Displayed is used to show counter output, which counts from 0 to 10, which is equal to A in HEX.
Counter will be reset in second last state.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
SerIn	Input	PIN_L22	5	B5_N1	PIN_L22	3.3-V LV...default)		24mA (default)	
SerOut	Output	PIN_Y21	6	B6_N1	PIN_Y21	3.3-V LV...default)		24mA (default)	
SerOutValid	Output	PIN_U22	6	B6_N1	PIN_U22	3.3-V LV...default)		24mA (default)	
clk	Input	PIN_L1	2	B2_N1	PIN_L1	3.3-V LV...default)		24mA (default)	
clkPB	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default)		24mA (default)	
out[6]	Output	PIN_D1	2	B2_N0	PIN_D1	3.3-V LV...default)		24mA (default)	
out[5]	Output	PIN_D2	2	B2_N0	PIN_D2	3.3-V LV...default)		24mA (default)	
out[4]	Output	PIN_G3	2	B2_N0	PIN_G3	3.3-V LV...default)		24mA (default)	
out[3]	Output	PIN_H4	2	B2_N0	PIN_H4	3.3-V LV...default)		24mA (default)	
out[2]	Output	PIN_H5	2	B2_N0	PIN_H5	3.3-V LV...default)		24mA (default)	
out[1]	Output	PIN_H6	2	B2_N0	PIN_H6	3.3-V LV...default)		24mA (default)	
out[0]	Output	PIN_E1	2	B2_N1	PIN_E1	3.3-V LV...default)		24mA (default)	
rst	Input	PIN_L2	2	B2_N1	PIN_L2	3.3-V LV...default)		24mA (default)	

Fig. 16 designed circuit pin planner schematic