交大密西根学院

UM-SJTU Joint Institute

Lab Report

4

UM-SJTU Joint Institute

Ve270 Introduction to Logic Design

by

Liu Yihao 515370910207

Ma ShiYao 515370910157

Date: 2017-06-20

Design of a Simple Counter

1 Objectives

To design a simple datapath using combinational building blocks in Xilinx ISE, and to implement the circuit in an FPGA chip.

2 Problem Definition

To form a Arithmetic and Logic Unit (ALU) which takes 4-bit input data and outputs the result(F and Cout). The algorithm being used is decided on the 2-input select signal. The algorithms are shown in Table 1.

S0	S1	F	Cout	Example
0	0	A+B	Carry out	A=1100, B=1110, F=1010, Cout=1
0	1	A−B	Carry out	A=1100, B=1110, F=1110, Cout=0
1	0	A and B	0	A=1100, B=1110, F=1100, Cout=0
1	1	A or B	0	A=1100, B=1110, F=1110, Cout=0

Table 1: Overall truth table

3 System Partitioning

1. 8 Switches and 2 select signal buttons. Switches are used to input two 4-bit binary input A and B. The buttos are used to select the algorithm to be executed. Then the input signals will be transmitted to the inside circuit to calculate and come up with the result.
2. Inside circuit. To receive the input signals and get the result. It will outputs some signals that controls the LED to illuminate.
3. LED. LED is used as the output indicator. The illumination of different LED can indicate different different binary digit as the output F and Cout.

4 Design Entry

The algorithm in this system was shown in Table 2.

Select signal		Algorithm	Output	
S0	S1		F	Cout
0	0	Add	A+B	Carry out
0	1	Subtract	A−B	Carry out
1	0	And	A and B	0
1	1	Or	A or B	0

Table 2: Algorithm table

According to the Table 2, we designed the ALU circuit, shown in Figure 1.

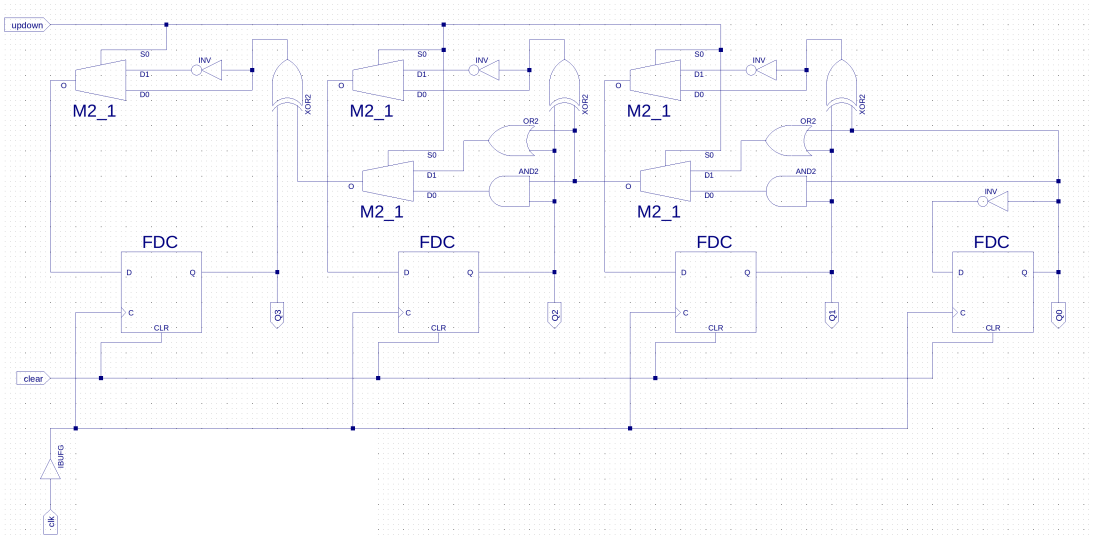

Figure 1: Schematics design of Counter

5 Test Plan

Test content	Test method
CA	Input signal that needs corresponding LED segment to illuminate, then observe if it illuminates. If it illuminates when corresponding signal inputted, this subsystem passes the test; otherwise it fails.
CB	
CC	
CD	
CE	
CF	
CG	
The overall system	Input every kind of the 4-bit binary digit signal: 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, then observe if LED displays the corresponding decimal digit:1,2,3,4,5,6,7,8,9. If so, the system passes the test; otherwise it fails.

6 Simulation Results

We simulated the result of the overall system with input values in Table 1, shown in Figure **??**. We found the values in the simulation identical to Table 1.

7 Conclusion

In this lab, we successfully finished it finally, but we also met with some problems in the process. Some existing modules in ISE will cause error in the simulation and pace process, such as the inverse-4 and mux-4, so we have to build a new module ourselves. We practiced build a ALU with 2-bit select input signal and four kinds of algorithms.

8 Appendix

The schematics of our design had been submitted to canvas before.
The verilog file is shown here:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    18:31:28 06/20/2017
// Design Name:
// Module Name:    counter_verilog
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module counter_verilog(Q, C, AN, updown, reset, clock);
    output [3:0] Q;
    output [6:0] C;
    output [3:0] AN;
    input updown, reset, clock;
    reg [3:0] Q = 4'b0000;
    ssd_driver ssd (C, Q);
    assign AN = 4'b1110;
    always @(posedge clock or posedge reset)
    begin
        if (reset) Q = 4'b0000;
        else if (updown) Q = Q - 1;
        else Q = Q + 1;
    end
endmodule

module ssd_driver(C, Q);
    output [6:0] C;
    input [3:0] Q;
    reg [6:0] C;
    always @(Q) begin
        case (Q)
            4'h0: C = 7'b1000000;
```

```
            4'h1: C = 7'b1111001;
            4'h2: C = 7'b0100100;
            4'h3: C = 7'b0110000;
            4'h4: C = 7'b0011001;
            4'h5: C = 7'b0010010;
            4'h6: C = 7'b0000010;
            4'h7: C = 7'b1111000;
            4'h8: C = 7'b0000000;
            4'h9: C = 7'b0010000;
            4'ha: C = 7'b0001000;
            4'hb: C = 7'b0000011;
            4'hc: C = 7'b1000110;
            4'hd: C = 7'b0100001;
            4'he: C = 7'b0000110;
            4'hf: C = 7'b0001110;
            default C = 7'b1111111;
        endcase
    end
endmodule
```

The verilog testbench file is shown here:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    18:50:15 06/20/2017
// Design Name:
// Module Name:    counter_test
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module counter_test();
    wire [3:0] Q, AN;
    wire [6:0] C;
    reg clock, reset, updown;
    counter_verilog UUT(Q, C, AN, updown, reset, clock);
    initial begin
```

```
        #0 clock = 1; updown = 0; reset = 0;
        #500 updown = 1;
        #1000 $stop;
    end
    always begin
        #10 clock = ~clock;
    end
    initial begin
        #300 reset = 1;
        #25 reset = 0;
    end
endmodule
```