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 FSM-Counter by both schematics and HDL modeling.

2 Problem Definition

To design a 4-bit up/down synchronous binary counter which is capable for 0 to 15 using 4 flip-flops sharing the same clock signal. It should be able to count up and down, controlled by an 1-bit inputting called "up/down" implemented by a switch. It is to increment when "up/down" signal is a binary 1 and to decrement if the signal is 0. There should be another input called "Reset" provided by a button. The counter should be 0 if the signal "Reset" is 1.

3 System Partitioning

1. Switches and buttons. Used to input signals like "Up/down mode choose", "Reset" and the signals to count.
2. Inside circuit. It is designed by using ISE and used to process the input signals and come up with output signals.
3. LED. The output the digits got by the circuit by illuminating different segments of LED. The rule of illumination according to the output digits is the same as Lab 2.

4 Design Entry

The function of this system was shown in Table 1.

Input signals		Output (Count)
Reset	Up/down	
1	X	0
0	0	Count+1
0	1	Count-1

Table 1: Function table

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

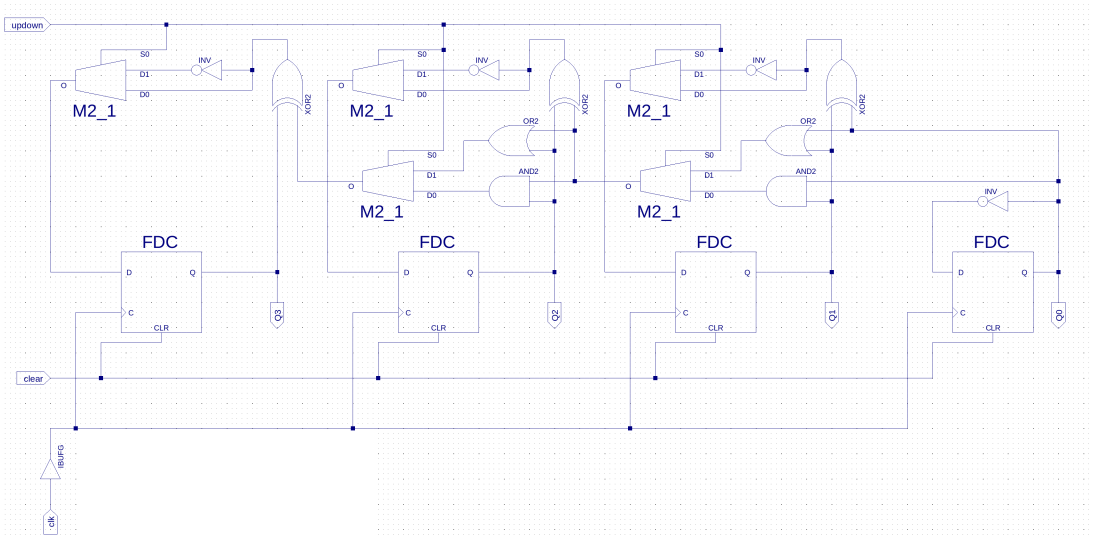

Figure 1: Schematics design of Counter

The rule of illumination according to the output (from Lab 2 SSD Driver) was shown in Figure 2 and Table 2.

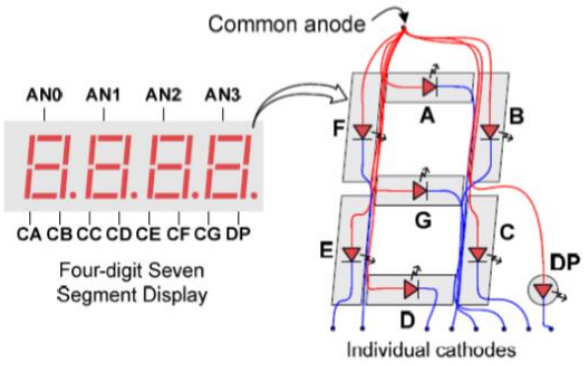

Figure 2: SSD Definition

Input	Outputs(LED segments)						
	CA	CB	CC	CD	CE	CF	CG
0000	0	0	0	0	0	0	1
0001	1	0	0	1	1	1	1
0010	0	0	1	0	0	1	0
0011	0	0	0	0	1	1	0
0100	1	0	0	1	1	0	0
0101	0	1	0	0	1	0	0
0110	0	1	0	0	0	0	0
0111	0	0	0	1	1	1	1
1000	0	0	0	0	0	0	0
1010	0	0	0	0	1	0	0
1011	1	1	0	0	0	0	0
1100	0	1	1	0	0	0	1
1101	1	0	0	0	0	1	0
1110	0	1	1	0	0	0	0
1111	0	1	1	1	0	0	0

Table 2: SSD truth table

5 Test Plan

Test content	Test method
Up Counter	Switch the counter on up/down mode, then input an “count” signal, observe the output digit.
Down Counter	If the output digit gets larger by 1 in “up” mode/smaller by 1 in “down” mode, it passes the test, otherwise it fails.
Reset button	Input an “reset” signal by the button, then observe the output digit. If the output digit becomes 0, it 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 **??**.

Figure 3: Simulation of Counter

We found the values in the simulation identical to Table **??**.

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
```