DSD F18

Lab 6 Complement and Hex Display

Name: Andrew Quick Date\_\_\_\_\_\_10/2/24\_\_\_\_\_\_\_\_

1. Explain why you might have unwanted display of segments that have been turned off?

**We’re not using them, don’t want to confuse the user or add unnecessary code.**

2. As an extra feature you can make you digits travel across the display. Explain how you would do that. Would you modify the sub module or the top module?

**To make digits travel across a 7-segment display, I would modify the top module by adding a counter or shift register to cycle through the digits and control which segment is active at each time step. The submodule for the 7-segment decoder would remain unchanged, as its function is purely to convert binary values to segment patterns. Timing control in the top module would handle the digit movement across the display.**

3. What would limit the number of digits that you could display by this time division multiplex method?

**The number of digits you can display using time-division multiplexing is limited by the refresh rate, which depends on how fast the multiplexing can switch between digits without visible flickering. If the switching speed is too slow, the display will appear to flicker. Additionally, hardware constraints like the clock speed and number of available output pins could also limit the number of digits.**

4. What method did you use to take the complement of the input byte?

**I used the two's complement method to take the complement of the input byte, which involves inverting the bits (~in) and then adding 1.**

5. There are high and low limits on the clock speed for multiplexing these displays. Give at least one consideration that provides a limit for the high side and at least one for the low side.

**For the high clock speed limit, the consideration is the response time of the 7-segment display; if the clock is too fast, the display won't have enough time to properly light up before switching to the next digit. On the low side, if the clock is too slow, human eyes will perceive flickering because each digit will not refresh fast enough, breaking the persistence of vision effect.**

6. Paste in your code and test bench

/// Need the Verilog code after testing to be added here

////// Test Bench////////////

`timescale 1ns/10ps

module tb\_counting\_buttons;

// Testbench signals

reg clk;

reg BTNC;

reg BTNU;

reg BTNR;

reg CPU\_RESETN;

reg [15:0] SW;

wire [7:0] anode;

wire [7:0] cathode;

// Clock generation

initial begin

clk = 0;

forever #5 clk = ~clk; // 100MHz clock (10ns period)

end

// DUT instantiation

counting\_buttons #(

.MODE("HEX"),

.NUM\_SEGMENTS(8),

.CLK\_PER(10),

.REFR\_RATE(1000),

.ASYNC\_BUTTON("SAFE")

) uut (

.clk(clk),

.BTNC(BTNC),

.BTNU(BTNU),

.BTNR(BTNR),

.CPU\_RESETN(CPU\_RESETN),

.SW(SW),

.anode(anode),

.cathode(cathode)

);

// Test scenario

initial begin

// Initialize inputs

CPU\_RESETN = 0;

BTNC = 0;

BTNU = 0;

BTNR = 0;

SW = 16'b0;

// Release reset

#20;

CPU\_RESETN = 1;

// Set input to 8-bit value (example: 8'b1111\_1010 = FA)

SW[7:0] = 8'b1111\_1010; // FA in hex

#20;

BTNU = 1; // Load the value into the display

#20;

BTNU = 0;

// Simulate pressing BTNR to display the 2's complement

#50;

BTNR = 1;

#20;

BTNR = 0;

// Observe the results on the 7-segment display

#200;

$finish;

end

endmodule