**Introduction:**

In this lab, we go through the complete FPGA design flow and design a stopwatch circuit on the Nexyts 3 Spartan-6 FPGA Board. The inputs of the stopwatch are buttons and slider switches. The buttons signals are debugged by debuggcors to avoid the noise to trigger the change multiple times.

The stopwatch will start as a basic clock which counts minutes and seconds. The left two digits of the seven segment displays minutes, and the left two seven-segment digits count seconds. Besides of the time counting, the stopwatch’s displaying time can be paused, or adjusted. Pause is control by a pause button. When the pause button is pressed, the time counting is frozen; when the pause button is pressed again, the time counting continue. Time adjusting is control by two slider switches. One is called ADJ and other one is called SEL. When the ADJ switch is on, the either the two minute digits or the two second digits will increment at a rate of 2 ticks per second, corresponding to the SEL slider switch equal to 0 an 1 respectively. When the stopwatch in on ADJ mode, the selected two digits also blink.

**Design:**

We divided this project into 4 modules, which are clk\_divider, anode\_selector, output\_selector, and display\_digit, debugcor, and stopwatch. The stopwatch module is the top module that connects all the instance of the sub other modules.

The following picture is the block diagram of the whole stopwatch design.

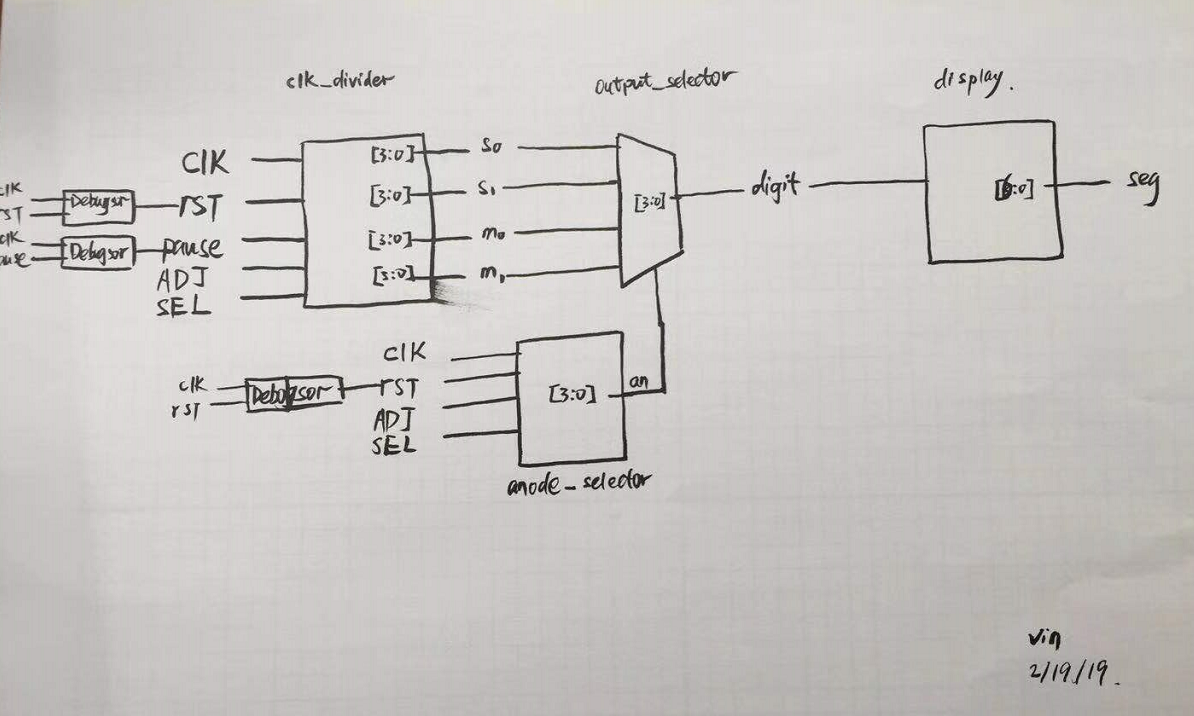


Figure 1.1. Block diagram of the stopwatch.

**Modules:**

1. **clk\_divider:**

The function of the clk\_divider is to generate the value of each 7-segment digit by dividing the built-in 100MHz clock and using counters. It has five inputs, which are clk, rst, pause, ADJ, and SEL. The clk is the 100MHz clock signal that will be used to generate 1Hz frequency clock. And the rst is the debugged reset button signal that will reset all 7-segment digits and counter to zero when its value is 1. Pause is the debugged pause button signal that stop the counting of the all counters when its value is 1. When the ADJ signal has value 1, the stopwatch is in a adjust mode. Either minute digits or second digits will change their incrementing frequency to with the SEL value equal to 0 or 1.

This module has four output, which are m0, m1, s0, s1. They are the digit value of the four seven-segment. The m0 and m1 display the minute state, and the s0 and s1 display the second state.

The following table illustrate the combination of the SEL and ADJ, and corresponding behavior of the four output values.

|  |  |  |  |
| --- | --- | --- | --- |
| ADJ | SEL | Minute (m0, m1) | Second (s0, s1) |
| 0 | 0 | Act normal | Act normal |
| 0 | 1 | Act normal | Act normal |
| 1 | 0 | Increment in 2Hz frequency | Act normal |
| 1 | 1 | Act normal | Increment in 2Hz frequency |

Table 1.1 ADJ, SEL signal and the corresponding behaviors of the stopwatch.

To make it easier to draw the schematic diagram, we divide this module into 4 parts, which are counter\_clk, s0, s1, m0, and m1.

The following are the schematic diagrams in this module.

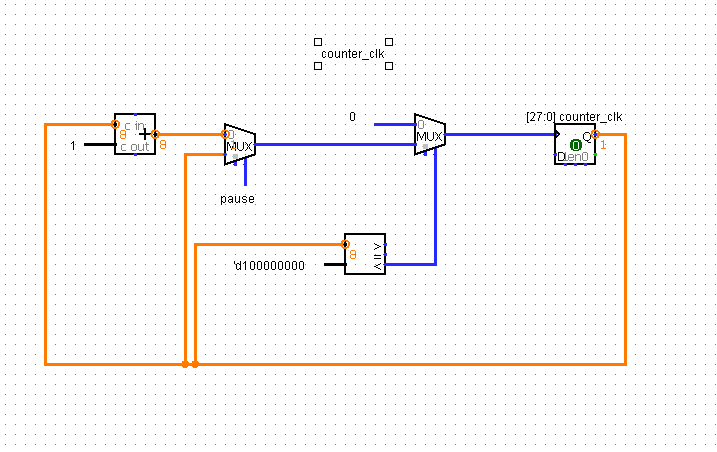


Figure 2.1 Schematic diagram of counter\_clk, a counter that use to divide the 100MHz clock to 1Hz.

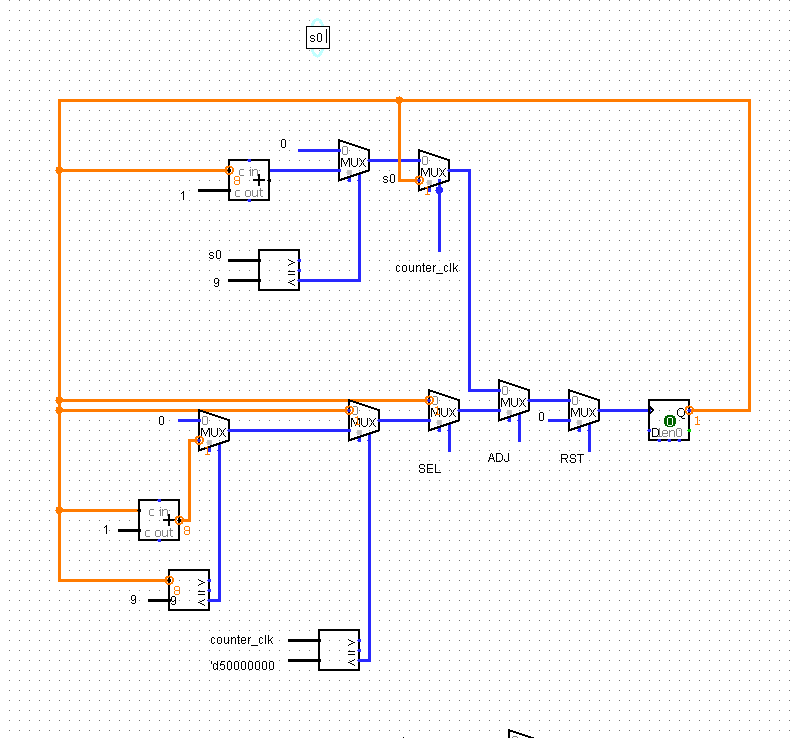


Figure 2.2 Schematic diagram of the s0 register/outpu.

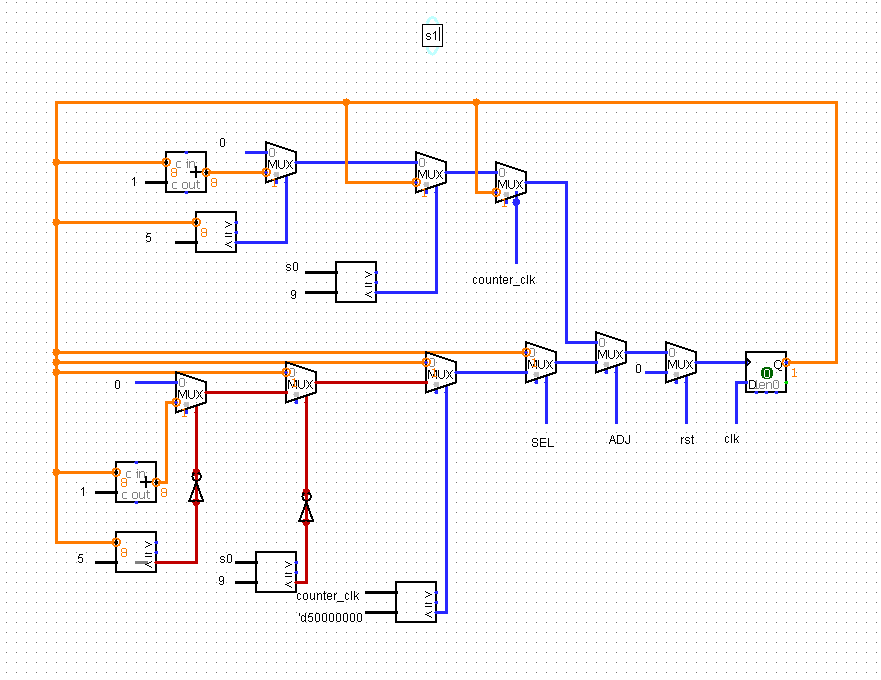


Figure 2.3 Schematic diagram of the s1 register/output.

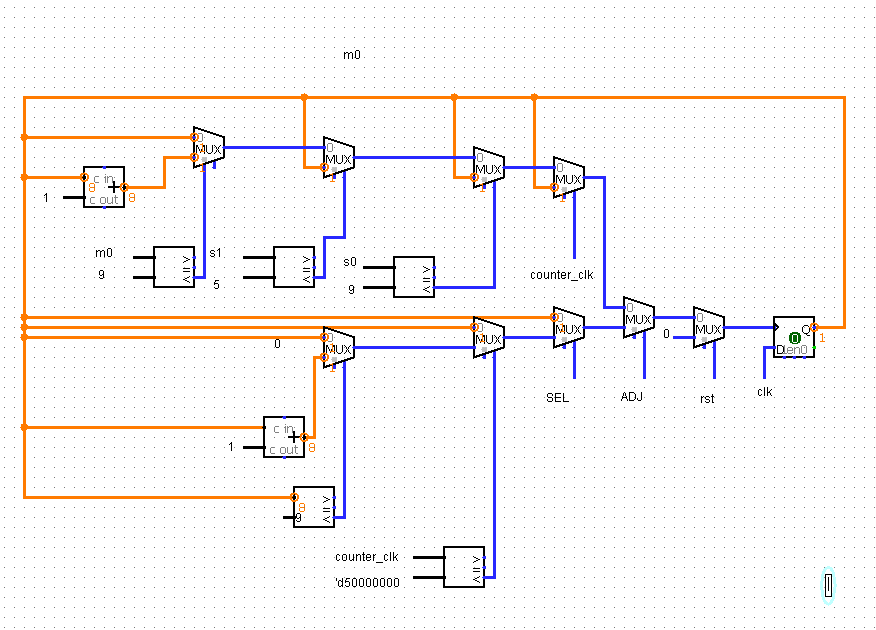


Figure 2.4 Schematic diagram of the m0 register/output.

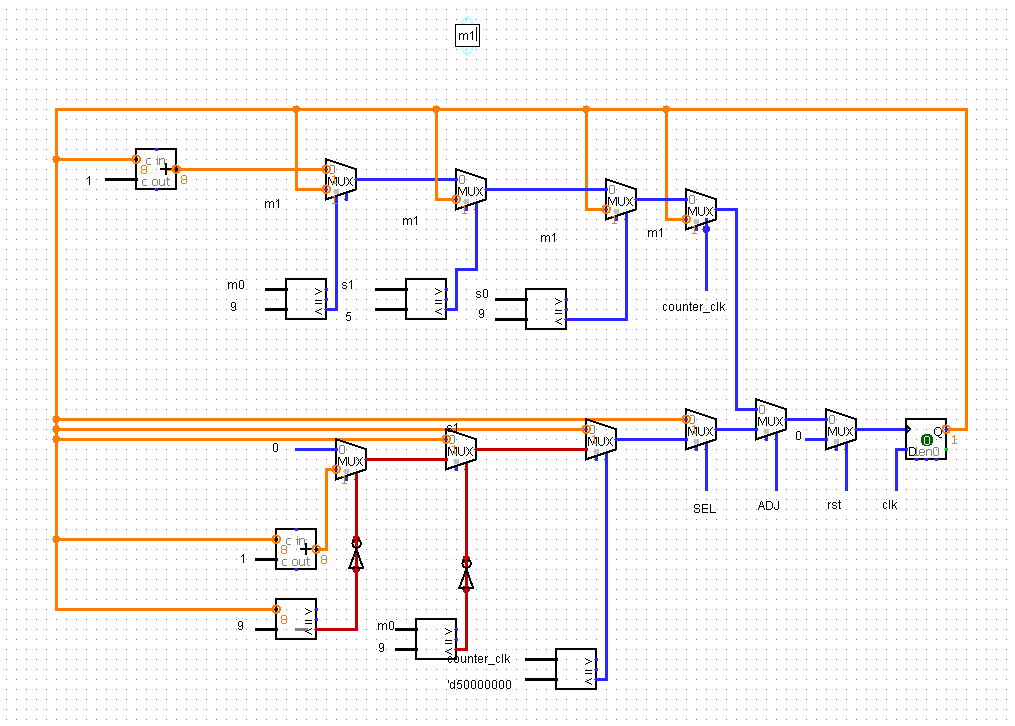


Figure 2.4 Schematic diagram of the m1 register/output.

1. **output\_selector:**

This module is a simple module that selects one input to the output. It is implemented by a multiplexer and some combinational logic circuit. The selector is the a 4-bit value of the anodes. The inputs of this module include, s0, s1, m0, m1, and an. The definitions of s0, s1, m0, m1 are explained in the clk\_divider module. The input an is a 4-bit value that tell the Multiplexer which anode is on. For example, when an[4:0] equal to 1110, the anode on the first seven-segment digit to the right is on, and other are turned off. Different from the normal multiplexer selector, the 4-bit selector only has 4 possibilities, which are 1110, 1101, 1011, 0111.

The following table illustrates how the multiplexer select the input to the output.

|  |  |
| --- | --- |
| Selector an [4:0] | Output digit [3:0] |
| 1110 | s0 |
| 1101 | s1 |
| 1011 | m0 |
| 0111 | m1 |

Table 2.2.1 The bit-values of the an[4:0] signal and their corresponding selected output.

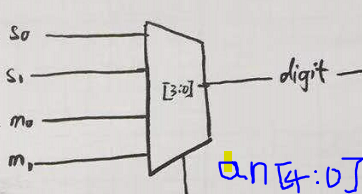


Figure 2.2.1 Diagram that explain how the s0, s1, m0, m1 are selected to the output digit.

Here is the schematic diagram of the output\_selector module.

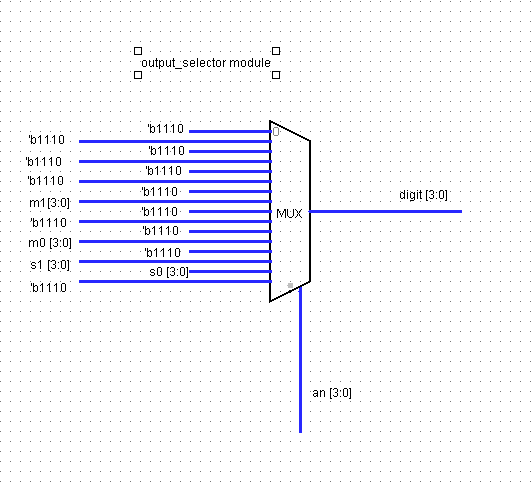


Figure 2.2.2 Schematic diagram of the output\_selector module.

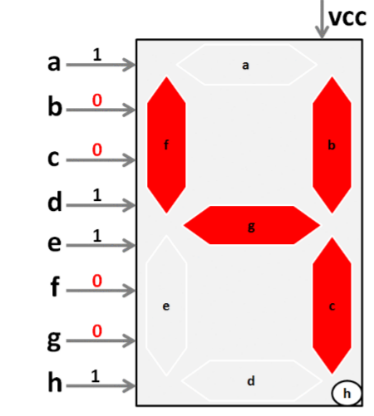
When an[3:0] = 1110, the 15th input is selected to the output, so the s0[3:0] is the 15th input of the multiplexer. In the similar way, s1[3:0], m0[3:0], and m1[3:0] are the 14th, 12th, and 8th input respectively.

The other inputs are set to ‘b1110, which is equal to 14 in decimal notation and will not be display in the display module.

1. **display:**

This module is a combinational module. It has one 4-bit input and one 7-bit output. Its input, digit [3:0], is the value of a digit, ranges 0 to 9. Its output, seg [6:0], is to be connected to the cathode pins of the led lights of the seven-segment digits. When this module receives a input signal of the digit value, it will turn the value of the seven cathode pin the desired combination that can display the digit of the input value to the seven-segment LED lights.

For example, when the input value is ‘d4, it will set the output seg [6:0] to 7'b0011001. Because the seg pin are cathode of the LED lights, when the value of one seg is 1, the LED is shut down; and when the seg is 0, the corresponding LED light has the opportunity to be lit up, depending on the value of the anode.

Figure 2.3.1 The seven-segment led light and the sequence of cathode value that display the number “4.”

This module on take care of the value of the cathode of the LED lights in the seven-segment digits. Because all four digits share the same cathode, when one digit is set to display “4”, all other digits’ cathood is set to display “4,” or shut down if their corresponding anode is set to 1.

Here is the schematic diagram of the display module:

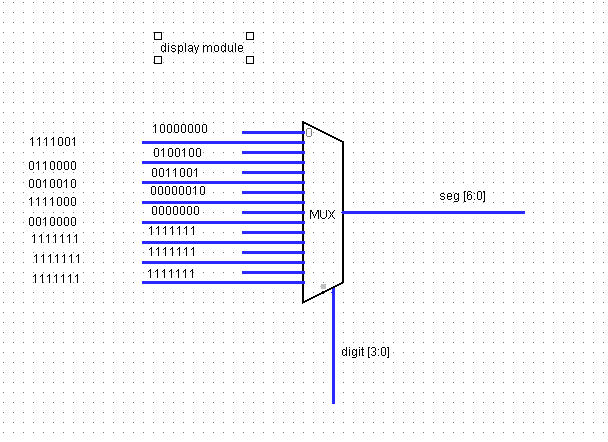


Figure 2.3.2 Schematic diagram of the display module the first ten inputs are the sequence of the bit-values of the cathodes of the seven-segment LED light that display digit from 0 to 9. The last six inputs are set to 1111111, which will turn off all the LED light.

1. **anode\_selector:**

This module is to control the frequency of displaying each digit. Because the 4 seven-segment share the same cathode, to display different number of different seven-segment digit, we need to have a relatively high frequency (about 200 Hz) to run through each digit. Because each digit’s LED lights are lit up and shut down at high frequency of the. Our eyes don’t aware of the discontinuous, and we are able to perceive different number of at a time.

Besides of running through the anode of the four digits at the relatively high frequency, the anode\_selector also generates a 3 Hz frequency that make the LED lights of the seven-segment number blink. The ADJ and SEL input determine which seven-segment digit to blink.

In short, the inputs of this module are clk, rst, ADJ, and SEL; its output is [3:0] an.

|  |  |  |  |
| --- | --- | --- | --- |
| ADJ | SEL | Minute (m0, m1) | Second (s0, s1) |
| 0 | 0 | Act normal | Act normal |
| 0 | 1 | Act normal | Act normal |
| 1 | 0 | BLINK at 3 Hz | Act normal |
| 1 | 1 | Act normal | BLINK at 3Hz |

­­­­Table 2.4.1 ADJ, SEL signal value and their corresponding behaviors about the Blinking.

Because the anode\_slector is a combination of sequential circuit and combinational circuit. We divide it into two part in the schematic diagram. As follows.

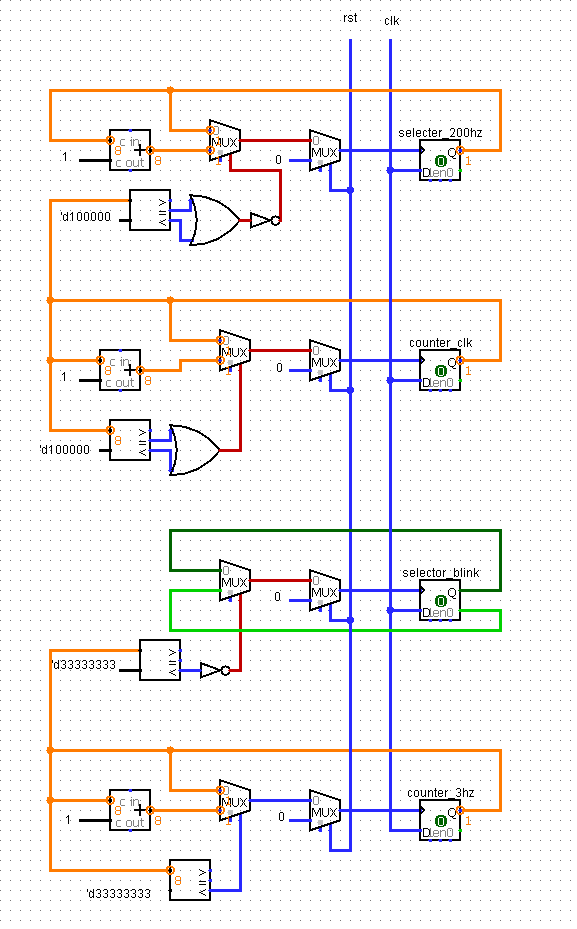


Figure 2.4.1 Schematic diagram of the sequential part of the anode\_selector. Selecter\_200hz, counter\_clk, selscotr\_blink, and counter\_3hz are register that store the value of divided clock of different frequency. They will be used in the combinational part of this module.

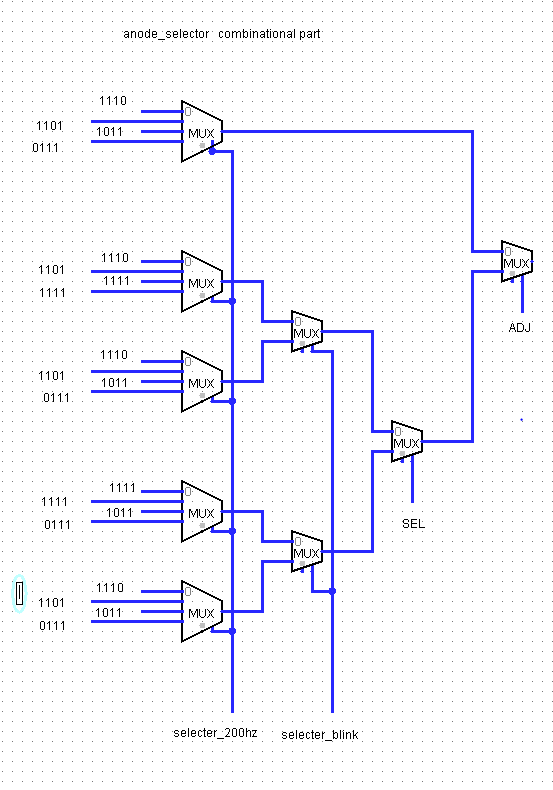


Figure 2.4.2 Combinational diagram of the sequential part of the anode\_selector. Selecter\_200hz, counter\_clk, selscotr\_blink, and counter\_3hz value are used to selected the inputs, which are the 4-bit sequence of the anode value to partially control the on/off of the LED lights

Review: I think we should have put selecter\_200hz, counter\_clk, selscotr\_blink into the clock\_divider module, so we could make the design and the module definition clearer. But we had been struggle with finding the way to use these clock signal. At first, we did put all these clocks in the clock\_divider module, and then use them in other blocks in this way:

always @ (posedge counter\_3hz)

But we later knew this isn’t the right way to do it. After several time of change, we figured it out and put these two sequential block and combinational block in the same modules.

1. **deboucer module**

This module is to sweep out the influence of noise. When we press a button, the signal generated by the button doesn’t not change to another stable value in a sudden. As the following figure shows, the signal will have several unstable changes before reaching to its relatively stable state. The deboucer will detect the value of the signal, which is either 0 or 1(low or high), in every clock period. If the number of high-state reaches to a certain number, then the machine should change the state of the button.

always @(posedge clk) begin

if (rst == 0) begin

btn\_count <= 0;

btn\_state <= 0;

end

else begin

if (btn\_count[10] == 1)begin

btn\_state <= ~ btin\_state;

btn\_count <= 0;

end

else begin

if (button)

btn\_count <= btn\_count + 1;

else

btn\_count <= btn\_count - 1;

end

end

end

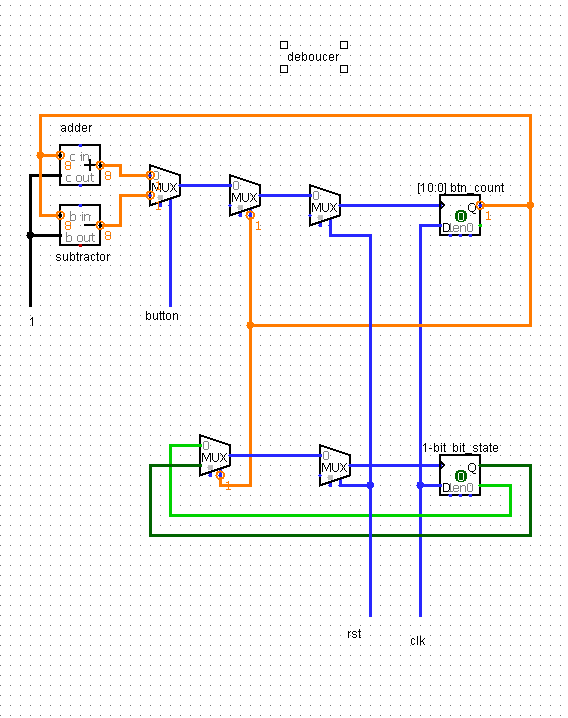


Figure 2.5 Schematic diagram of the debouncer module. Instead of taking the signal of a button as an input, we use a 11-bit to count the number of clock cycle when the value of the button signal is high.The value of the btn\_count increment by one when the button value is 1, and decrement by 1 when the button signal is 0, so we used adder and subtractor to implement this module. When the value of btn\_count[0] reach to 1, the value of but\_state reverses.

Review: The debouncer module was is only module that we are not confident with. We finally finished it with the help of the TA. In my dobouncer module, the rst(reset) signal is used to initial the value of the btn\_count and btn\_state. Different from what you taught me, I insist that I should reverse the btn\_state instead of just setting it to 1 when btn\_coutn[10] equal to 1. In this way, we press the pause button the second time will restart the stopwatch.

**Summary:**

In this lab, we have learned how to display digits on the seven-segment digits. We learned the debouncor and build a small project of the combination of sequential circuits and combinational circuits. Also, it is also interesting to design how to divide a program into several modules. Besides, I corrected many Verilog syntax error. It gets much harder to draw the schedemic diagram when the program become large.