EEE339 Assignment 1

Student Name:Yang Li

Student Number: 1715977

Abstract

The assignment of this experiment was to design a digital clock based on the Verilog language using the Quartus II software, using modular programming ideas. The code was written and then simulated and debugged using the Quartus II software. Finally, hardware tests were carried out on a DE1 board (Altera Cyclone II 2C20 FPGA).

Contents

Design Objectives

Design of an electronic clock based on the Verilog hardware programming language using Quartus II software and running on a DE1 board with the following basic requirements.

- basic timing display function, i.e. minutes and seconds on 4 seven-segment tubes, range (00:00 - 59:59).

- time setting function with adjustable minutes and seconds.

- Stopwatch function with 4 seven-segment tubes sub-metered to display seconds and tenths of a second.

Function Description

Switch SW9 is the key position to control the clock system on and off, when SW9 is high (1) the system is on, when SW9 is low (0) the system is reset and all functions are suspended.

- Clock timing function: When SW9 is high, SW0 and SW1 are low to enter the clock timing function. If the clock has previously used the time setting function, the clock start time will be timed from the set time.

- Time setting function: When SW9 and SW0 are high and SW1 is low enter the time setting function of the clock. In this function, the four digits of minutes and seconds can be adjusted in any way. Firstly, the position of the adjustment is selected by KEY1, the starting default position is the second digit of the second (in order to facilitate the user to confirm the position of the adjustment, the selected position will be blinked by 2HZ), secondly, the digit of the selected position is set, KEY2 is plus, KEY3 is minus.

- Stopwatch function: When SW9 and SW1 are high and SW0 is low enter the stopwatch function. KEY2 is the reset signal to reset the four digits of the stopwatch.

System overview with a diagram

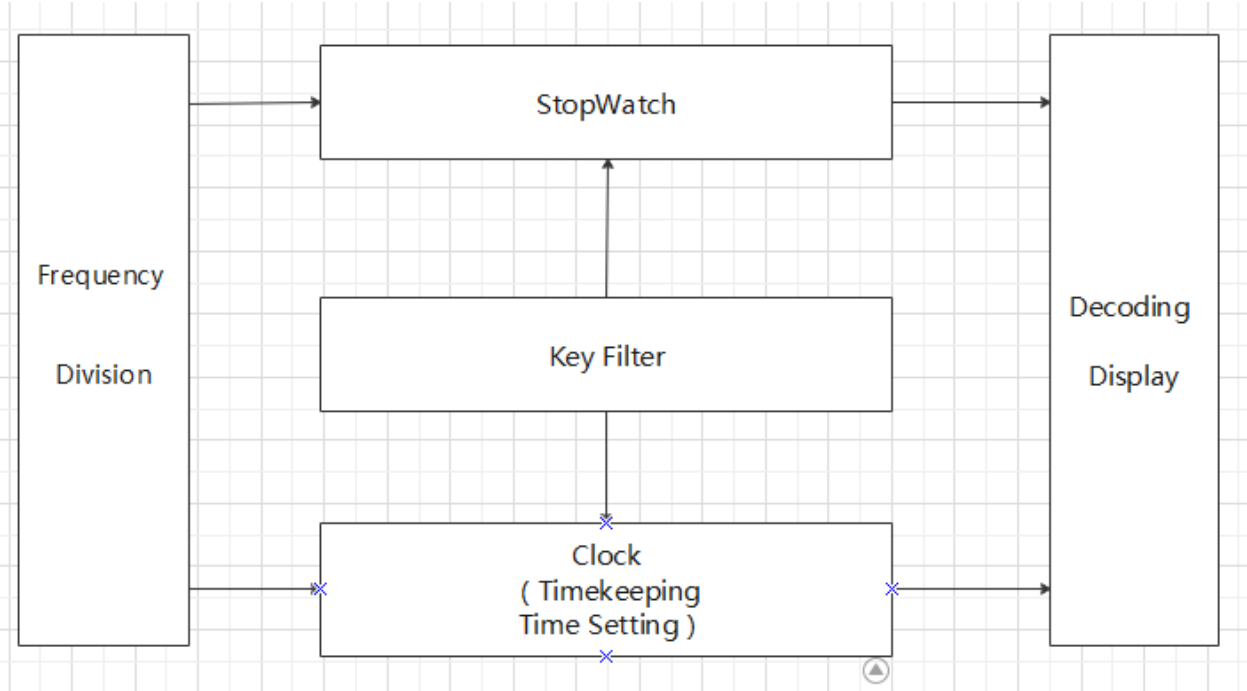

Diagram 1: Schematic Diagram of Digital Clock

The overall part of the digital clock can be divided into five basic modules according to the purpose of the design: the Frequency Divider module, the Digital Clock module, the Stopwatch module, the Key Filter module, the Decoding Scan Display module. The Diagram 1 shows the relationship between each module.

Sythesised circuits

a. Frequency Divider Module

The main function of the frequency division module: to provide 1HZ pulse signal for the clock timing, to provide 100HZ pulse signal for the stopwatch timing, to provide 1KHZ pulse signal for the decoding display function.

```
always@(posedge clk)
begin

    count1<=count1+1;
    if(count1<25_000_000)          //2
        clk_1HZ<=1;
    else if(count1<49_999_999)     //4
        clk_1HZ<=0;
    if(count1==49_999_999)         //4
        count1<=0;
end
```

In the Verilog code design by counting to achieve frequency division. This experiment access to the system clock frequency is 50KHZ. Counter to count until the count to half of the required number of frequencies in the 50KHZ pulse signal, and then the signal is flipped. For example, a 1HZ signal pulse signal needs to be counted first 25_000_000, then the signal is inverted and counted to 49_999_999 and the counter is cleared. Figure 2 shows the precess of generating a 1Hz frequency.

Figure 2: Verilog Code at 1Hz Frequency

b. Digital Clock Module

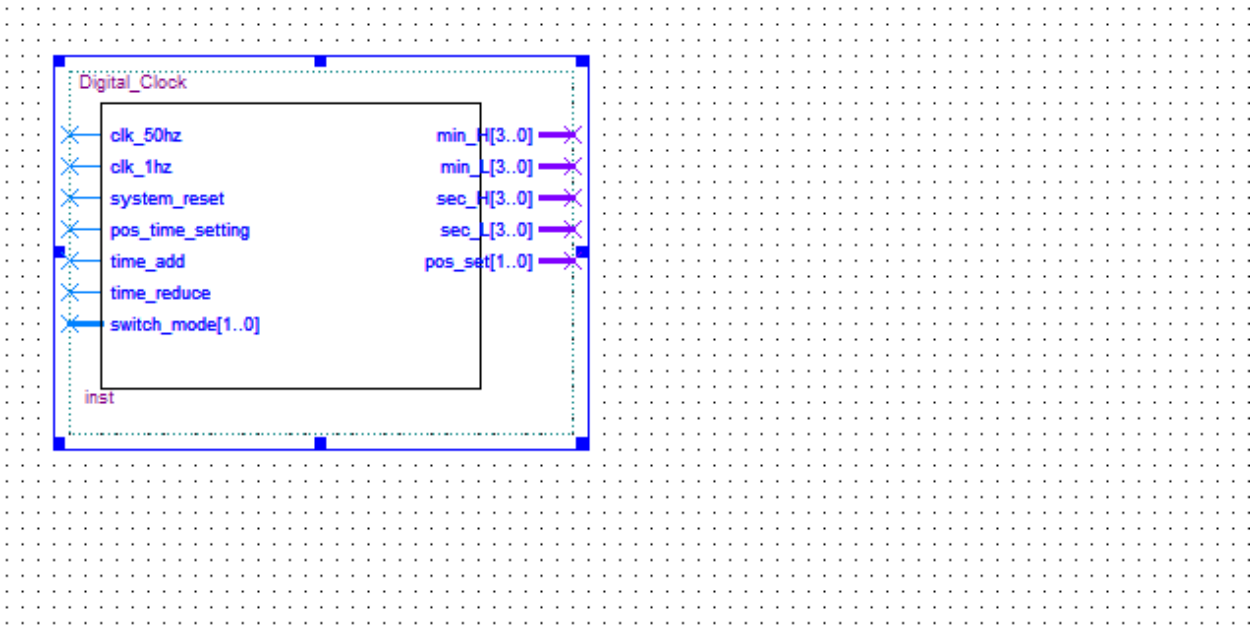

Figure 3: The Sythesised Circuit of Digital Clock Module

(1) Digital Clock Timing Function

Firstly, two 4-bit registers(Sec_L, Sec_H) are used to record the digits of the second and the tens of the second. Secondly, a pulse signal of 1HZ is obtained through the frequency divider module to access this function. With a 1 Hz pulse, the first digit of the second is counted until 10 (binary 1010), the tenth digit of the second is cleared to zero. When the tens digit of the second is 6 (binary 0110), the seconds digit and the tens digit of the second are cleared to zero and then the minutes digit is rounded up. The case of seconds is the same as that of minutes.

(2) Time Setting Function

Firstly, four 4-bit registers (temp_sec_H, temp_sec_L, temp_min_H, temp_min_L) are used to record each of the four digits of the clock timing. Secondly four 4 registers are defined to record the time setting digits (sec_H_set, sec_L_set, min_H_set, min_L_set) and a 2-bit register (pos_set) is used to record the position of the time setting.

When the digital clock enters the time setting function, the four registers that record the clock timing assign the stored value to the four registers that record the time setting. Then the value of 'pos_set' is used to determine the position of the time setting, and finally to determine whether the user presses the add (KEY2) or reduce key (KEY3). When the time setting is completed, the values stored in the four time setting registers are assigned to the four registers of the clock timing.

c. Stopwatch Module

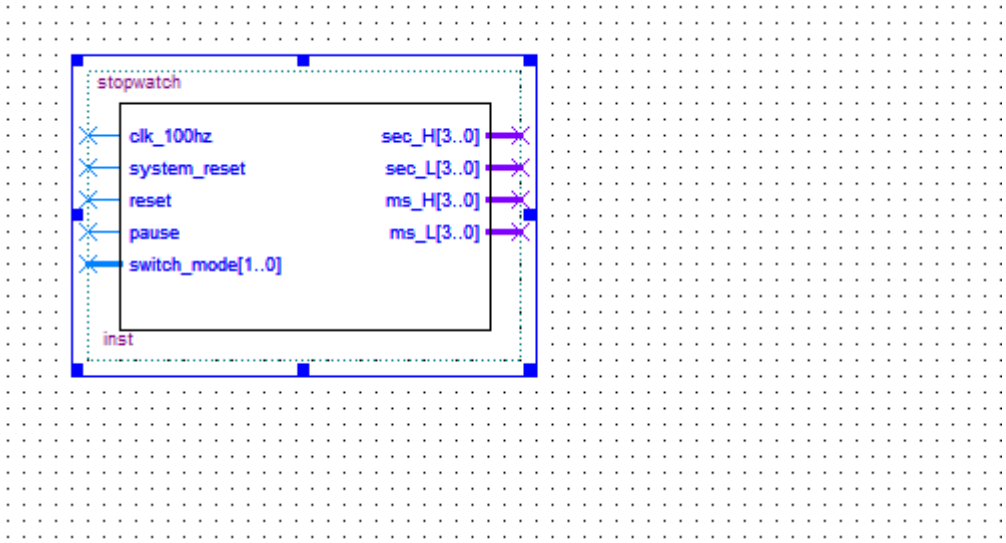

Figure 4: The Sythesised Circuit of Stopwatch Module

The principle of the stopwatch module is similar to that of digital clock timing. Four registers (sec_H,sec_L,ms_H,ms_L) are used to record the four digits of the stopwatch.

For illustration purposes, the tens and single digits of the seconds and the tens and single digits of the tenths of a second are replaced by m1, m2, m3, m4. At a pulse signal of 100 Hz, m4 starts counting. When m4 is full of tens, m3 is rounded up by 1 and m4 is cleared. m2 is the same as m3 and m4, and when 2 is full of tens, m1 is rounded up by 1 and m2 is cleared. When m1 is 6, m1 to m4 are all cleared.

Pause:

When SW3 is high, the four registers are assigned the same value, i.e. ms_L <= ms_L.

Reset:

When the clear button is triggered, the four registers are assigned a value of 0.

d. Key Filter Module

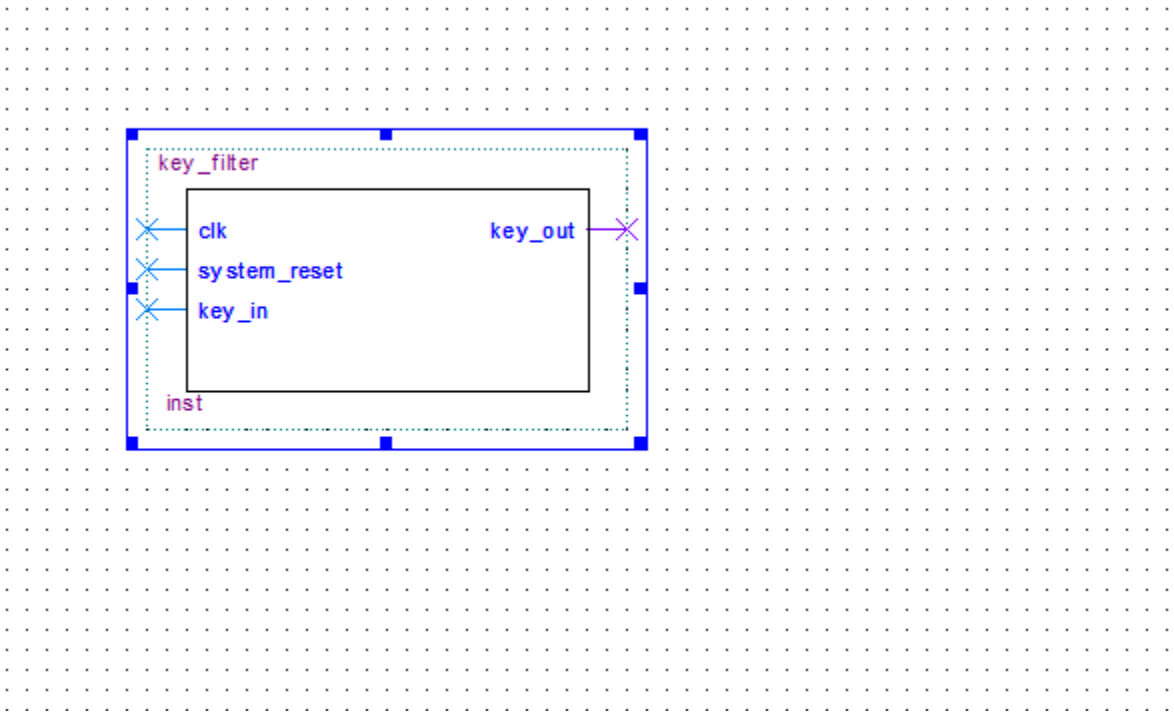

Figure 5: The Sythesised Circuit of Key Filter Module

The key filter used in this design uses the delayed filter principle. When a key change is detected, a delay time of 20ms is waited for the jitter to disappear and then the key state is detected again. If the detected state is the same as the previous key state, the module outputs the key trigger signal.

e. Decoding Scan Display Module

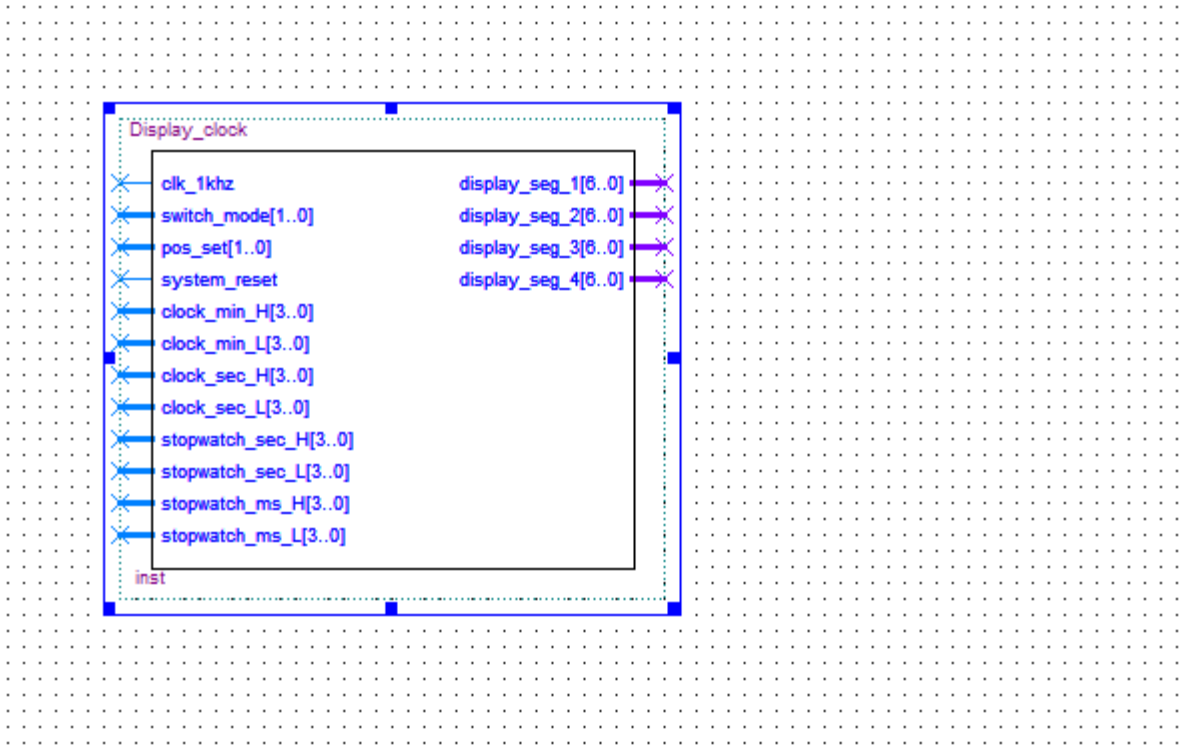

Figure 6: The Sythesised Circuit of Decoding Scan Display Module

(1) Display function

The display module uses a high frequency scan display. Firstly, defining four four-bit registers (display_1,2,3,4) will record what the digital clock system wants to display. Secondly four seven-bit registers are defined to compile the corresponding values of the display (display_1,2,3,4) into the segment code. Finally, the values of the 4 digital tubes are displayed in a constant loop at a frequency of 1kHz.

(2) Time Setting blinking function

When the time setting function is recognised as being entered, the position of the time setting is first confirmed by the value of ‘pos_set’. Next, a counter is defined to count from 0 when the scan cycle enters the time set position. When the count is less than 500 the display will show the current number. When the count is greater than 500 and less than 1000 the digital tubes are all unlit. The counter is cleared when the count is greater than 1000.

f. Top Module

Instantiate each of the above modules in a top-level module, linking each module together.

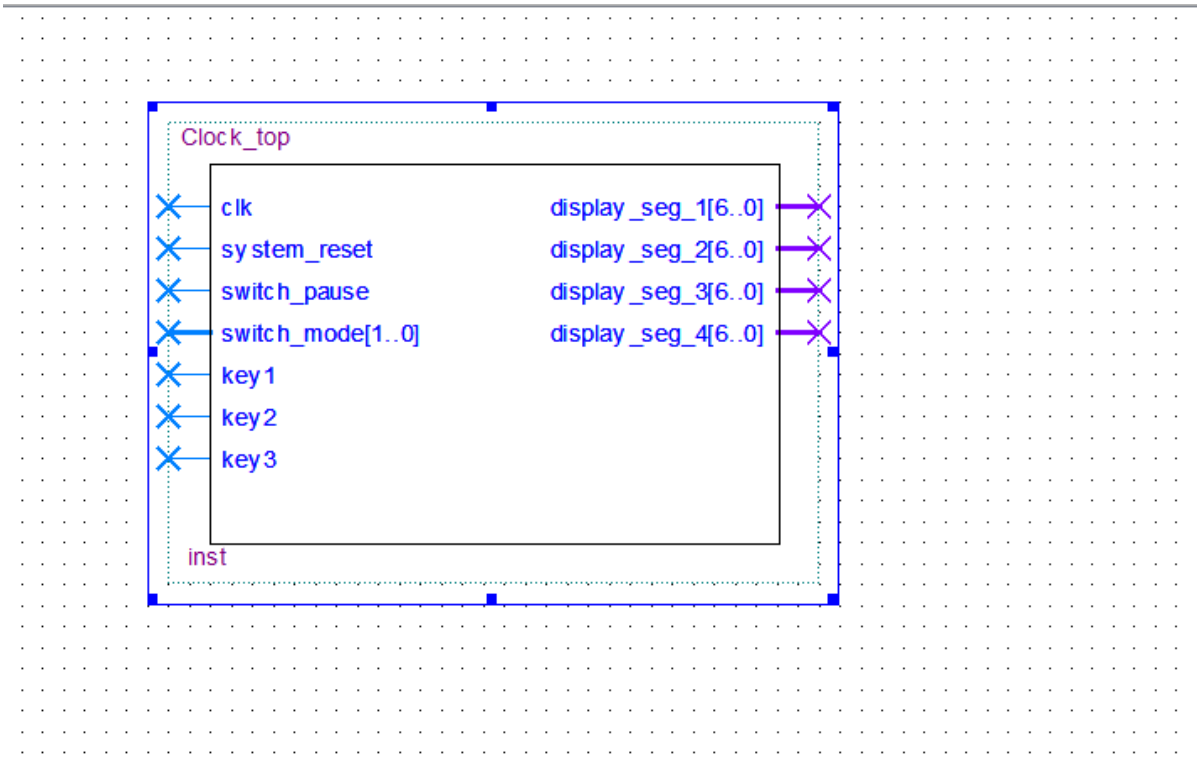

Figure 7: The Sythesised Circuit of Top Module

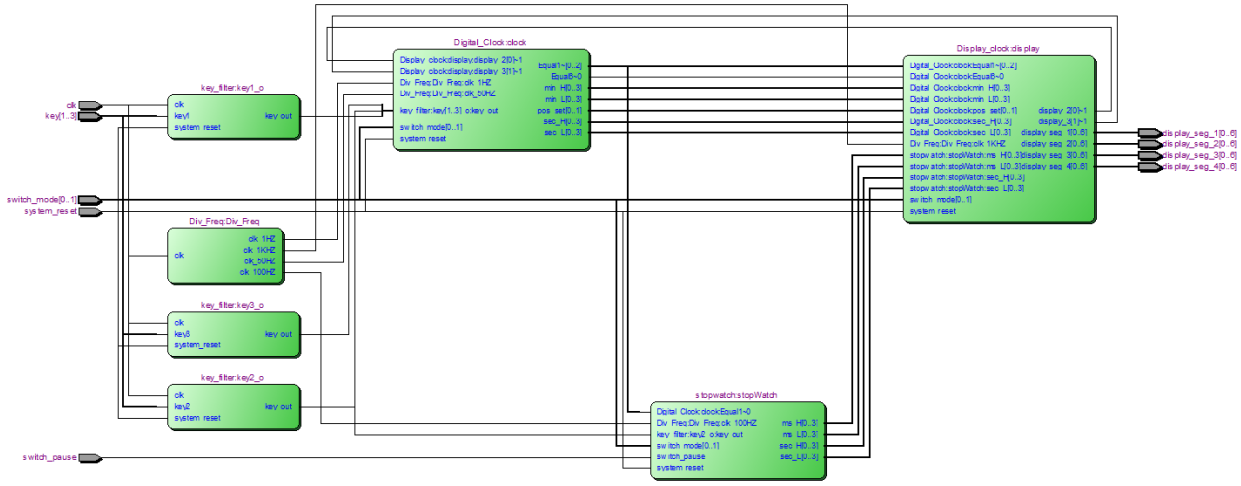

Figure 8: The Circuit of Digital Clock

Simulation of the core components

a. Frequency Division Function

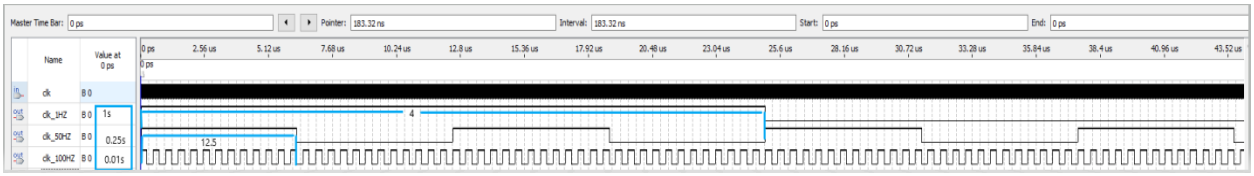

Figure 9: Simulation results of Frequency Division Function

Figure 9 shows the simulation results of the frequency divider function, according to the figure you can see the relationship between 1Hz and 4Hz and 100Hz. According to the simulation results it can be seen that 2 cycles of 4Hz are equal to half a 1Hz cycle and 12.5 cycles of 100Hz are equal to half a 4Hz cycle.

b. Digital Clock Timing Function

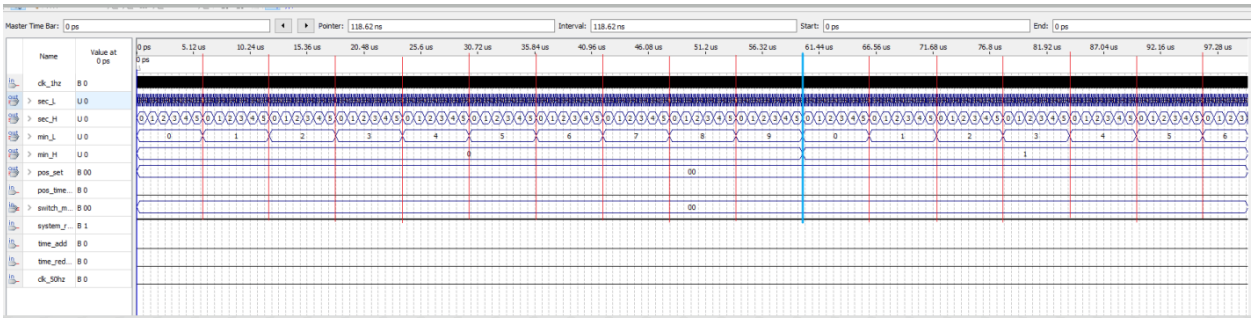

Figure 10: Simulation results of Digital Clock Timing Function

The red line in Figure 10 represents the marker points per minute. Where the blue line represents the ten-minute marker point.

c. Time Setting Function

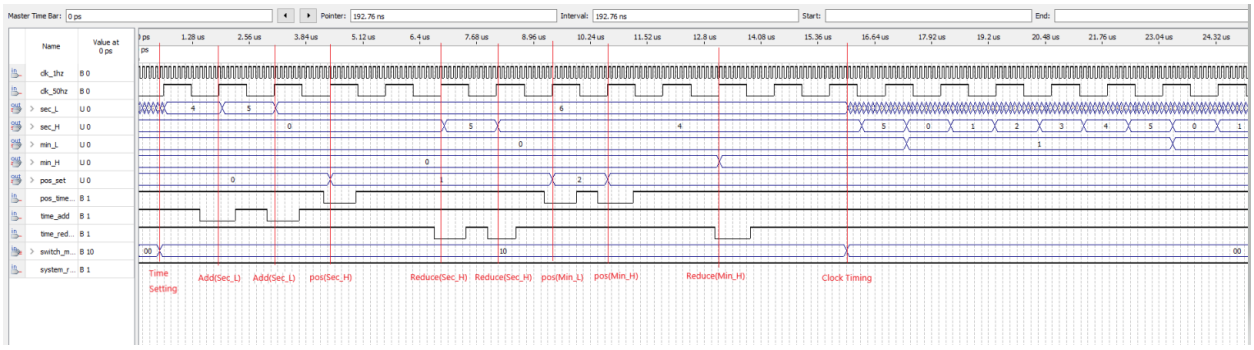

Figure 11: Simulation results of Timing Setting Function

In Figure 11, ‘pos_set’ indicates the position of the time setting. The red line in the diagram marks the content and time point of each operation. According to the figure it can be seen that when the module is switched from the time setting function to the clock timing function, the digits per bit are adjusted according to the time setting function.

d. Stopwatch Function

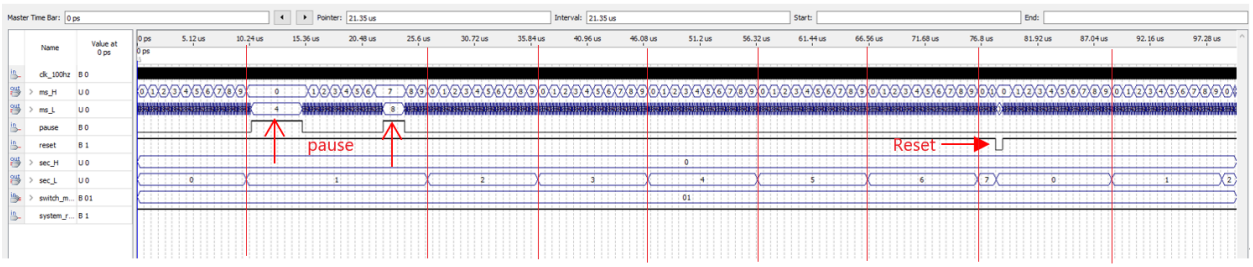

Figure 12: Simulation results of Stopwatch Function

The red line in Fig. 12 represents the single digit progression of seconds plus one, and marks the point in time when the pause and reset operations are performed.

e. Decoding Scan Display Function

Figure 13: Simulation results of Decoding Scan Display Function

The blue color in Figure 13 represents the four 7-bit segment codes output. The simulation result of ‘display_seg_1’ flashing imitation can be seen in the Time Setting interval. (‘7F’ means that none of the digital tubes are lit)

f. Key Filter Function

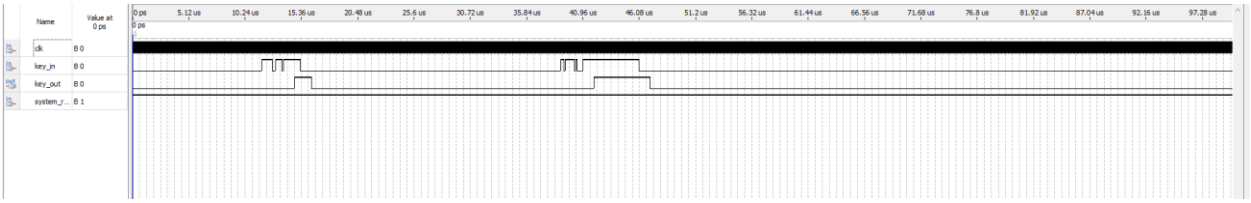

Figure 14: Simulation results of Key Filter Function

The simulation in Figure 14 mimics the effect of key jitter, and according to the diagram the effectiveness of the key jitter module can be seen. Only after a key press has been sustained for a certain period of time is the key validity signal output.

Conclusions

In this assignment, a digital clock was successfully designed using Verilog HDL, completing all the basic functions. During this design, the importance of code specification was appreciated. This is because during the writing process, the problems with begin and end and the semicolon ';' at the end of the statement caused many compilation failures and took some time to fix these bugs.

Secondly, the importance of designing the framework of the module was recognised. In this assignment, the module was designed with the clock timing function and the time setting function in one module, which reduced the extensibility and flexibility and portability of the code. This problem made it more difficult to add a countdown timer and an alarm clock function later on.

Appendix

Clock_top.v

```
/*
        Student Name: Yang Li
        Studen Number: 1715977

        SW9 DOWN ---- System Reset
        SW9 UP---- System Start
                                                  ___________
                                ________________
        ___________________
        Digital Clock   |       SW9 UP      |                    StopWatch   |
              SW9 UP                  |                    Time Set        |
   SW9 UP                         |
                                                |   SW0 DOWN    |
                                        |       SW0 UP                  |
                                          |           SW0 DOWN                |
                                                |   SW1 DOWN    |
                                        |       SW1 DOWN            |
                                      |           SW1 UP                      |
                                                _____
   _                                                    |           Function    :
            |                                                   |           Function:
                        |
                                                                  |   SW2 UP(Pause)   |
                                                |       KEY1(Position)          |

                                                                  |   KEY2 (Reset)
       |                                                          |       KEY2(Add Time
     |
                                                                   _______
   _                                                  |   KEY3(Reduce Time)   |

                                                                                  ______
   ___
*/


module Clock_top(clk,
                                                    system_reset,
                                                    switch_pause,
                                                    switch_mode,
                                                    key1,
                                                    key2,
                                                    key3,
                                                    display_seg_1,
                                                    display_seg_2,
                                                    display_seg_3,
                                                    display_seg_4,

```

```
                                                    );

input                                       clk;
input                                       system_reset,
                                                    switch_pause,
                                                    key1,
                                                    key2,
                                                    key3;
input[1:0]                          switch_mode;
//display
output[6:0]                 display_seg_1,
                                                    display_seg_2,
                                                    display_seg_3,
                                                    display_seg_4;
//clock frequency
wire                                        clk_1HZ,
                                                    clk_50HZ,
                                                    clk_100HZ,
                                                    clk_1KHZ;

wire[1:0]                           pos_set;


//key output
wire                                        key1_out,
                                                    key2_out,
                                                    key3_out;
//clock_time
wire[3:0]                           clock_min_H,
                                                    clock_min_L,
                                                    clock_sec_H,
                                                    clock_sec_L;
//stopwatch
wire[3:0]                           stopwatch_sec_H,
                                                    stopwatch_sec_L,
                                                    stopwatch_ms_H,
                                                    stopwatch_ms_L;


key_filter key1_o(              .clk                                        (clk),
                                                                            .system_reset       (system_reset),
                                                                            .key_in             (key1),
                                                                            .key_out            (key1_out)
);

key_filter key2_o(              .clk                                        (clk),
                                                                            .system_reset       (system_reset),
                                                                            .key_in             (key2),
                                                                            .key_out            (key2_out)
);

```

```
key_filter key3_o(              .clk                                    (clk),
                                                                        .system_reset
            (system_reset),
                                                                        .key_in
                                (key3),
                                                                        .key_out
                        (key3_out)
);

Div_Freq Div_Freq(
                                                                        .clk
                                (clk),
                                                                        .clk_1HZ
                        (clk_1HZ),
                                                                        .clk_50HZ
                    (clk_50HZ),
                                                                        .clk_100HZ
                (clk_100HZ),
                                                                        .clk_1KHZ
                    (clk_1KHZ)
                                                                        );
stopwatch stopWatch(
                                                                        .clk_100hz
                (clk_100HZ),
                                                                        .system_reset
            (system_reset), //switch9
                                                                        .reset
                            (key2_out), //key2
                                                                        .pause
                        (switch_pause), //switch 2
                                                                        .switch_mode
             (switch_mode),
                                                                        .sec_H
                         (stopwatch_sec_H),
                                                                        .sec_L
                         (stopwatch_sec_L),
                                                                        .ms_H
                         (stopwatch_ms_H),
                                                                        .ms_L
                         (stopwatch_ms_L)
                                                                        );


Display_clock display   (
                                                                .clk_1khz
                     (clk_1KHZ),
                                                                .switch_mode
             (switch_mode),
                                                                .pos_set
                         (pos_set),
                                                                .system_reset
             (system_reset),
                                                                .clock_min_H
             (clock_min_H),
                                                                .clock_min_L
                 (clock_min_L),
                                                                .clock_sec_H
                 (clock_sec_H),
```

```
                                                                .clock_sec_L
                  (clock_sec_L),
                                                                .stopwatch_sec_H
      (stopwatch_sec_H),
                                                                .stopwatch_sec_L
          (stopwatch_sec_L),
                                                                .stopwatch_ms_H (
   stopwatch_ms_H),
                                                                .stopwatch_ms_L
      (stopwatch_ms_L),

                                                                .display_seg_1
             (display_seg_1),
                                                                .display_seg_2
             (display_seg_2),
                                                                .display_seg_3
             (display_seg_3),
                                                                .display_seg_4
             (display_seg_4)
                                                                );


Digital_Clock clock (
                                                                .clk_50hz
                     (clk_50HZ),
                                                                .clk_1hz
                         (clk_1HZ),
                                                                .system_reset
             (system_reset),
                                                                .pos_time_setting
      (key1_out),//key1
                                                                .time_add
                     (key2_out), //key2
                                                                .time_reduce
                 (key3_out), //key3
                                                                .switch_mode
             (switch_mode),
                                                                .min_H
                         (clock_min_H),
                                                                .min_L
                         (clock_min_L),
                                                                .sec_H
                         (clock_sec_H),
                                                                .sec_L
                         (clock_sec_L),
                                                                //modify
                                                                .pos_set
                         (pos_set)
                                                                //over
                                                            );


endmodule
```

Div_Freq.v

```
/*
         Student Name: Yang Li
         Studen Number: 1715977
*/
module Div_Freq (clk,
                                                clk_1HZ,
                                                clk_50HZ,
                                                clk_100HZ,
                                                clk_1KHZ);
input                                       clk;
output                                   clk_1HZ,
                                                clk_50HZ,
                                                clk_100HZ,
                                                clk_1KHZ;
reg                                            clk_1HZ,
                                                clk_50HZ,
                                                clk_100HZ,
                                                clk_1KHZ;
integer                                  count1 = 0,
                                                count2 = 0,
                                                count100 = 0,
                                                count1k = 0;


always@(posedge clk)
begin

        count1<=count1+1;
        if(count1<25_000_000)                    //25_000_000
                clk_1HZ<=1;
        else if(count1<49_999_999)       //49999999
                clk_1HZ<=0;
        if(count1==49_999_999)              //49999999
                count1<=0;
 end
always@(posedge clk)
begin//0.4s
            count2<=count2+1;
        if(count2<625)                    //1250000       10_000_000(0.4s)
                clk_50HZ<=1;
        else if(count2<12_49)        //2499999       19_999_999
                clk_50HZ<=0;
        if(count2==12_49)               //2499999       19_999_999
                count2<=0;
end
always@(posedge clk)
begin
       count100<=count100+1;

        if(count100<25)//250000
                clk_100HZ<=1;
        else if(count100<49)//499999
                clk_100HZ<=0;
        if(count100==49)//499999
                count100<=0;
end


 always@(posedge clk)
begin
        count1k<=count1k+1;
```

```
62.
63.         if(count1k<25)
64.                 clk_1KHZ<=1;
65.         else if(count1k<49)
66.                 clk_1KHZ<=0;
67.         if(count1k==49)
68.                 count1k<=0;
69. end
70. endmodule
```

stopwatch.v

```
1.  /*
2.          Student Name: Yang Li
3.          Studen Number: 1715977
4.  */
5.  module stopwatch(clk_100hz,
6.                                                  system_reset,
7.                                                  reset, //key2
8.                                                  pause, //switch sw2
9.                                                  switch_mode,
10.                                                 sec_H,
11.                                                 sec_L,
12.                                                 ms_H,
13.                                                 ms_L);
14. input                                       clk_100hz;
15.
16. input                                       reset,
17.                                                 pause,
18.                                                 system_reset;
19. input [1:0 ]                        switch_mode;
20. output [3:0]                    sec_H,
21.                                                 sec_L,
22.                                                 ms_H,
23.                                                 ms_L;
24.
25. reg [3:0]                               sec_H,
26.                                                 sec_L,
27.                                                 ms_H,
28.                                                 ms_L;

29.
30. always @(posedge clk_100hz )
31. begin
32.             if(system_reset ==0)
33.             begin
34.                             ms_L <= 4'b0000;
35.                             ms_H <= 4'b0000;
36.                             sec_H <= 4'b0000;
37.                             sec_L <= 4'b0000;
38.             end
39.
40.             else
41.             begin
42.                             if(switch_mode == 2'b01)
43.                             begin
44.                                             if(~reset)
45.                                             begin
46.                                                             ms_L <= 4'b0000;
```

```
                                                ms_H <= 4'b0000;
                                                sec_H <= 4'b0000;
                                                sec_L <= 4'b0000;
                                        end

                                        else if(pause)
                                        begin
                                                ms_L <= ms_L;
                                                ms_H <= ms_H;
                                                sec_H <= sec_H;
                                                sec_L <= sec_L;

                                        end

                                        else
                                        begin
                                                ms_L <= ms_L + 4'b0001;
                                                if(ms_L == 4'b1001)
                                                begin
                                                        ms_L <= 4'b0000;
                                                        ms_H <= ms_H+ 4'b0
    001;
                                                end

                                                if(ms_L == 4'b1001 && ms_H ==
    4'b1001)
                                                begin
                                                        ms_H <= 4'b000;
                                                        sec_L <= sec_L + 4
    'b0001;
                                                end

                                                if(sec_L == 4'b1001 &&(ms_H ==
     4'b0101 && ms_L == 4'b1001))
                                                begin
                                                        sec_L <= 4'b0000;

                                                        sec_H <= sec_H + 4
    'b0001;
                                                end

                                                if( (sec_H == 4'b0101 &&  sec_
    L == 4'b1001) &&  (ms_H == 4'b0101 && ms_L == 4'b1001)    )
                                                begin
                                                        ms_L <= 4'b0000;
                                                        ms_H <= 4'b0000;
                                                        sec_H <= 4'b0000;

                                                        sec_L <= 4'b0000;

                                                end
                                        end
                                end
                        end
                end
        end
end

endmodule
```

Digital_Clock.v

```
/*
         Student Name: Yang Li
         Studen Number: 1715977
*/
module Digital_Clock(clk_50hz,
                                                    clk_1hz,
                                                    system_reset,
                                                    pos_time_setting,//key 1
                                                    time_add,  //key2
                                                    time_reduce,  //key3
                                                    switch_mode,
                                                    min_H,
                                                    min_L,
                                                    sec_H,
                                                    sec_L,
                                                    //modify
                                                    pos_set);
                                                    //over

input                                           clk_50hz,
                                                    clk_1hz;

input                                          system_reset;

input                                           pos_time_setting,
                                                    time_add,
                                                    time_reduce;

input [1:0]                                 switch_mode;

output [3:0]                            min_H,
                                                    min_L,
                                                    sec_H,
                                                    sec_L;

reg [3:0]                                   min_H,
                                                    min_L,
                                                    sec_H,
                                                    sec_L;

reg[3:0]                                    min_H_set,
                                                    min_L_set,
                                                    sec_H_set,
                                                    sec_L_set;

reg [3:0]                                   temp_min_H,
                                                    temp_min_L,
                                                    temp_sec_H,
                                                    temp_sec_L;
//modidy
output[1:0]                             pos_set;
//over
```

```
reg[1:0]                                    pos_set;

reg                                             loop;
reg[3:0]                                      conflict;

//clock count
always@(posedge clk_1hz)
begin
        if(system_reset == 0)
        begin
                    min_H <= 4'b0000;
                    min_L<= 4'b0000;
                    sec_H  <= 4'b0000;
                    sec_L   <= 4'b0000;
                    temp_sec_H  <=       4'b0000;
                    temp_sec_L  <=       4'b0000;
                    temp_min_H  <=       4'b0000;
                    temp_min_L  <=   4'b0000;

        end
        else begin
        if(switch_mode == 2'b10)
        begin
                        sec_H <= sec_H_set;
                        sec_L <= sec_L_set;
                        min_H <= min_H_set;
                        min_L <= min_L_set;
        end

        //else if(switch_mode == 2'b00)
        else if(switch_mode == 2'b00)
        begin
                        sec_L <= sec_L + 4'b0001;

                        if(sec_L == 4'b1001)
                        begin
                                    sec_L <= 4'b0000;
                                    sec_H <= sec_H + 4'b0001;
                        end

                        if(sec_H == 4'b0101 && sec_L == 4'b1001)
                        begin
                                    sec_H  <= 4'b0000;
                                    min_L <= min_L + 4'b0001;
                        end

                        if(min_L == 4'b1001 &&(sec_H == 4'b0101 && sec_L == 4'b1001))
                         begin
                                     min_L <= 4'b0000;
                                     min_H <= min_H + 4'b0001;
                         end
                         if((min_H >= 4'b0101 && min_L >= 4'b1001) &&(sec_H >= 4'b0101 && sec_L >= 4'b1001))
                         begin
                                     min_H <= 4'b0000;
                                     min_L<= 4'b0000;
                                     sec_H  <= 4'b0000;
                                     sec_L   <= 4'b0000;
                         end

```

```
                        temp_sec_H  <=       sec_H;
                        temp_sec_L  <=       sec_L;
                        temp_min_H  <=       min_H;
                        temp_min_L  <=       min_L;


            end
end
end
//_____________________________________________

//_____________________________________________
//time_setting
always@(posedge clk_50hz )//posedge clk_50hz )//or posedge pos_time_setting o
    r )
begin
                if(system_reset == 0)
                begin
                                    sec_H_set <= 4'b0000;
                                    sec_L_set   <= 4'b0000;
                                    min_H_set <= 4'b0000;
                                    min_L_set <= 4'b0000 ;
                                    pos_set     <= 2'b00;


                end

            else
            begin

                    if(switch_mode == 2'b10)
                            begin
                            if(loop == 0)
                            begin
                                        sec_H_set <= temp_sec_H;
                                        sec_L_set   <= temp_sec_L;//+ 4'b0001
    ; // to solve bug (when switch the mode to time_set, the value of sec_L will r
    educe 1 )
                                        min_H_set <= temp_min_H;
                                        min_L_set <= temp_min_L ;
                                        loop <= loop +1;

                            end
                // determine the position which is changed


                            if(~pos_time_setting)

                            begin
                                        if(pos_set == 2'b11)
                                        begin
                                                    pos_set <= 2'b00;
                                        end
                                        else
                                                    pos_set <= pos_set+2'b01;

                            end
                            // second_LOW - time setting
                            if(pos_set==2'b00)
                            begin
                                        if(~time_add)
```

```
169.
170.                                         begin
171.                                                 if(sec_L_set == 4'b1001)

172.                                                 begin
173.                                                         sec_L_set <=
    4'b0000;
174.                                                 end
175.                                                 else
176.                                                         sec_L_set <=
    sec_L_set + 4'b0001;
177.                                                 end
178.                                         if(~time_reduce)
179.
180.                                                 begin
181.                                                 if(sec_L_set == 4'b0000)

182.                                                 begin
183.                                                         sec_L_set <=
    4'b1001;
184.                                                 end
185.                                                 else
186.                                                         sec_L_set <=
    sec_L_set - 4'b0001;
187.
188.                                                 end
189.                                 end
190.                                 //second_HIGH -time setting
191.                                 if(pos_set==2'b01)
192.                                 begin
193.                                         if(~time_add)
194.
195.                                         begin
196.                                                 if(sec_H_set == 4'b0101)

197.                                                 begin
198.                                                         sec_H_set <=
    4'b0000;
199.                                                 end
200.                                                 else
201.                                                         sec_H_set <=
    sec_H_set + 4'b0001;
202.                                 end
203.
204.                                                 if(~time_reduce)

205.                                                 begin
206.                                                 if(sec_H_set == 4'b0000)

207.                                                 begin
208.                                                         sec_H_set <=
    4'b0101;
209.                                                 end
210.                                                 else
211.                                                         sec_H_set <=
    sec_H_set - 4'b0001;
212.
213.                                 end
214.                                 end
215.                                         //min_LOW -time setting
216.                                         if(pos_set==2'b10)
```

```
                                        begin

                                            if(~time_add)

                                            begin
                                                if(min_L_set == 4'b1001)
                                                begin
                                                    min_L_set <= 4'b0000;
                                                end
                                                else
                                                    min_L_set <= min_L_set + 4'b0001;
                                            end

                                            if(~time_reduce)

                                            begin
                                                if(min_L_set == 4'b0000)
                                                begin
                                                    min_L_set <= 4'b1001;
                                                end
                                                else
                                                    min_L_set <= min_L_set - 4'b0001;

                                            end
                                        end
                                        //min_HIGH - time setting
                                        if(pos_set == 2'b11)
                                        begin

                                            if(~time_add)

                                            begin
                                                if(min_H_set == 4'b0101)
                                                begin
                                                    min_H_set <= 4'b0000;
                                                end
                                                else
                                                    min_H_set <= min_H_set + 4'b0001;
                                            end

                                                if(~time_reduce)

                                            begin
                                                if(min_H_set == 4'b0000)
                                                begin
                                                    min_H_set <= 4'b0101;
                                                end
                                                else

                                                    min_H_set <= min_H_set - 4'b0001;
```

```
264.
265.                                                     end
266.
267.                                             end
268.                                             //finish
269.
270.                 end
271.     else if(switch_mode == 2'b00)
272.     begin
273.                 loop <= 0;
274.     end
275.
276.     end
277.end
278.
279.
280.endmodule
```

Display_clock.v

```
1.  /*
2.          Student Name: Yang Li
3.          Studen Number: 1715977
4.  */
5.  module Display_clock(clk_1khz,
6.                                                          switch_mode,
7.                                                          //modify
8.                                                          pos_set,
9.                                                          system_reset,
10.                                                         //over
11.                                                         clock_min_H,
12.                                                         clock_min_L,
13.                                                         clock_sec_H,
14.                                                         clock_sec_L,
15.                                                         stopwatch_sec_H,
16.                                                         stopwatch_sec_L,

17.                                                         stopwatch_ms_H,
18.                                                         stopwatch_ms_L,

19.                                                         display_seg_1,
20.                                                         display_seg_2,
21.                                                         display_seg_3,
22.                                                         display_seg_4
23.                                                     );
24. input                                               clk_1khz;
25. input                                               system_reset;
26. input[1:0]                                  switch_mode;
27. //modify
28. input[1:0]                                      pos_set;
29. //over
30. input[3:0]                                  clock_min_H,
31.                                                         clock_min_L,
32.                                                         clock_sec_H,
33.                                                         clock_sec_L,
34.                                                         stopwatch_sec_H,
35.                                                         stopwatch_sec_L,
```

```
                                                            stopwatch_ms_H,
                                                            stopwatch_ms_L;

output [6:0]                                display_seg_1,
                                                            display_seg_2,
                                                            display_seg_3,
                                                            display_seg_4;

reg [6:0]                                           display_seg_1,
                                                            display_seg_2,
                                                            display_seg_3,
                                                            display_seg_4;

reg [3:0]                                           display_1,
                                                            display_2,
                                                            display_3,
                                                            display_4;

reg [1:0]                                       loop_clock,
                                                            loop_stopwatch,
                                                            loop_time_set;
integer                                             CNT;

always@(posedge clk_1khz)
begin
if(system_reset ==0)
begin
                display_seg_1 =  7'b0000000;
                display_seg_2 =  7'b0000000;
                display_seg_3 =  7'b0000000;
                display_seg_4 =  7'b0000000;
end

else
begin
            if(switch_mode == 2'b01)
            begin
                        display_4 <= stopwatch_sec_H;
                        display_3 <=stopwatch_sec_L;
                        display_2 <=stopwatch_ms_H;
                        display_1 <= stopwatch_ms_L;

                        case(loop_stopwatch)
                                    2'b00:
                                    begin
                                                case(display_1)
                                                            4'b0000 : display_
    seg_1 = 7'b1000000;
                                                            4'b0001 : display_
    seg_1 = 7'b1111001;
                                                            4'b0010 : display_
    seg_1 = 7'b0100100;
                                                            4'b0011 : display_
    seg_1 = 7'b0110000;
                                                            4'b0100 : display_
    seg_1 = 7'b0011001;
                                                            4'b0101 : display_
    seg_1 = 7'b0010010;
                                                            4'b0110 : display_
    seg_1 = 7'b0000010;
```

```
                                                    4'b0111 : display_
    seg_1 = 7'b1111000;
                                                    4'b1000 : display_
    seg_1 = 7'b0000000;
                                                    4'b1001 : display_
    seg_1 = 7'b0010000;
                                                    default   : displa
    y_seg_1 = 7'b1111111;
                                                endcase
                                        end
                                        2'b01:
                                        begin
                                                case(display_2)
                                                    4'b0000 : display_
    seg_2 = 7'b1000000;
                                                    4'b0001 : display_
    seg_2 = 7'b1111001;
                                                    4'b0010 : display_
    seg_2 = 7'b0100100;
                                                     4'b0011 : display
    _seg_2 = 7'b0110000;
                                                    4'b0100 : display
    _seg_2 = 7'b0011001;
                                                    4'b0101 : display
    _seg_2 = 7'b0010010;
                                                    4'b0110 : display
    _seg_2 = 7'b0000010;
                                                    4'b0111 : display
    _seg_2 = 7'b1111000;
                                                    4'b1000 : display
    _seg_2 = 7'b0000000;
                                                    4'b1001 : display
    _seg_2 = 7'b0010000;
                                                    default : display
    _seg_2 = 7'b1111111;
                                                endcase
                                        end
                                        2'b10:
                                        begin
                                                case(display_3)
                                                    4'b0000 : display
    _seg_3 = 7'b1000000;
                                                    4'b0001 : display
    _seg_3 = 7'b1111001;
                                                    4'b0010 : display
    _seg_3 = 7'b0100100;
                                                    4'b0011 : display
    _seg_3 = 7'b0110000;
                                                    4'b0100 : display
    _seg_3 = 7'b0011001;
                                                    4'b0101 : display
    _seg_3 = 7'b0010010;
                                                    4'b0110 : display
    _seg_3 = 7'b0000010;
                                                    4'b0111 : display
    _seg_3 = 7'b1111000;
                                                    4'b1000 : display
    _seg_3 = 7'b0000000;
                                                    4'b1001 : display
    _seg_3 = 7'b0010000;
```

```
                                                                default : display_seg_3 = 7'b1111111;
                                                        endcase
                                        end
                                        2'b11:
                                        begin
                                                case(display_4)
                                                                4'b0000 : display_seg_4 = 7'b1000000;
                                                                4'b0001 : display_seg_4 = 7'b1111001;
                                                                4'b0010 : display_seg_4 = 7'b0100100;
                                                                4'b0011 : display_seg_4 = 7'b0110000;
                                                                4'b0100 : display_seg_4 = 7'b0011001;
                                                                4'b0101 : display_seg_4 = 7'b0010010;
                                                                4'b0110 : display_seg_4 = 7'b0000010;
                                                                4'b0111 : display_seg_4 = 7'b1111000;
                                                                4'b1000 : display_seg_4 = 7'b0000000;
                                                                4'b1001 : display_seg_4 = 7'b0010000;
                                                                default : display_seg_4 = 7'b1111111;
                                                endcase
                                        end

                        endcase

                        loop_stopwatch <= loop_stopwatch + 2'b01;
                        if(loop_stopwatch == 2'b11)
                        begin
                                        loop_stopwatch <= 2'b00;
                        end
            end
            //clock
            else if( switch_mode == 2'b00)
            begin
                        display_4 <=     clock_min_H;
                        display_3 <= clock_min_L;
                        display_2 <=     clock_sec_H;
                        display_1 <= clock_sec_L;

                        loop_clock <= loop_clock + 2'b01;
                        if(loop_clock == 2'b11)
                        begin
                                        loop_clock <= 2'b00;
                        end

                        case(loop_clock)
                                        2'b00:
                                        begin
                                                case(display_1)
                                                                4'b0000 : display_seg_1 = 7'b1000000;
```

```
                                                        4'b0001 : display_seg_1 = 7'b1111001;
                                                        4'b0010 : display_seg_1 = 7'b0100100;
                                                        4'b0011 : display_seg_1 = 7'b0110000;
                                                        4'b0100 : display_seg_1 = 7'b0011001;
                                                        4'b0101 : display_seg_1 = 7'b0010010;
                                                        4'b0110 : display_seg_1 = 7'b0000010;
                                                        4'b0111 : display_seg_1 = 7'b1111000;
                                                        4'b1000 : display_seg_1 = 7'b0000000;
                                                        4'b1001 : display_seg_1 = 7'b0010000;
                                                        default   : display_seg_1 = 7'b1111111;
                                                    endcase
                                                end
                                                2'b01:
                                                begin
                                                    case(display_2)
                                                        4'b0000 : display_seg_2 = 7'b1000000;
                                                        4'b0001 : display_seg_2 = 7'b1111001;
                                                        4'b0010 : display_seg_2 = 7'b0100100;
                                                        4'b0011 : display_seg_2 = 7'b0110000;
                                                        4'b0100 : display_seg_2 = 7'b0011001;
                                                        4'b0101 : display_seg_2 = 7'b0010010;
                                                        4'b0110 : display_seg_2 = 7'b0000010;
                                                        4'b0111 : display_seg_2 = 7'b1111000;
                                                        4'b1000 : display_seg_2 = 7'b0000000;
                                                        4'b1001 : display_seg_2 = 7'b0010000;
                                                        default : display_seg_2 = 7'b1111111;
                                                    endcase
                                                end
                                                2'b10:
                                                begin
                                                    case(display_3)
                                                        4'b0000 : display_seg_3 = 7'b1000000;
                                                        4'b0001 : display_seg_3 = 7'b1111001;
                                                        4'b0010 : display_seg_3 = 7'b0100100;
                                                        4'b0011 : display_seg_3 = 7'b0110000;
```

```
                                                        4'b0100 : display_seg_3 = 7'b0011001;
                                                        4'b0101 : display_seg_3 = 7'b0010010;
                                                        4'b0110 : display_seg_3 = 7'b0000010;
                                                        4'b0111 : display_seg_3 = 7'b1111000;
                                                        4'b1000 : display_seg_3 = 7'b0000000;
                                                        4'b1001 : display_seg_3 = 7'b0010000;
                                                        default : display_seg_3 = 7'b1111111;
                                                    endcase
                                            end
                                            2'b11:
                                            begin
                                                    case(display_4)
                                                                4'b0000 : display_seg_4 = 7'b1000000;
                                                                4'b0001 : display_seg_4 = 7'b1111001;
                                                                4'b0010 : display_seg_4 = 7'b0100100;
                                                                4'b0011 : display_seg_4 = 7'b0110000;
                                                                4'b0100 : display_seg_4 = 7'b0011001;
                                                                4'b0101 : display_seg_4 = 7'b0010010;
                                                                4'b0110 : display_seg_4 = 7'b0000010;
                                                                4'b0111 : display_seg_4 = 7'b1111000;
                                                                4'b1000 : display_seg_4 = 7'b0000000;
                                                                4'b1001 : display_seg_4 = 7'b0010000;
                                                                default : display_seg_4 = 7'b1111111;
                                                        endcase
                                            end

                        endcase


                end

            //time_set
                else if(switch_mode == 2'b10 )
                begin
                                //display_4 <=   clock_min_H;
                                //display_3 <=   clock_min_L;
                                //display_2 <=   clock_sec_H;
                                //display_1 <=   clock_sec_L;
                                loop_time_set <= loop_time_set + 2'b01;
                                if(loop_time_set == 2'b11)
                                begin
                                            loop_time_set <= 2'b00;
                                end
```

```
248.
249.                         if(pos_set == 2'b00)
250.                         begin
251.                                         display_4 <=     clock_min_H;
252.                                         display_3 <= clock_min_L;
253.                                         display_2 <=     clock_sec_H;
254.
255.                                         if(CNT <500)
256.                                         begin
257.                                                         display_1 <= clock_sec_L;
258.                                                         CNT <= CNT +1;
259.                                         end
260.
261.                                         else if(CNT >=500 && CNT<1000)
262.                                         begin
263.                                                         CNT <= CNT +1;
264.                                                         display_1 <= 4'b1111;
265.                                         end
266.                                         else if( CNT == 1000)
267.                                                         CNT <= 0;
268.
269.                         end
270.
271.                         else if(pos_set == 2'b01)
272.                         begin
273.                                         display_4 <=     clock_min_H;
274.                                         display_3 <= clock_min_L;
275.                                         display_1 <= clock_sec_L;
276.
277.                                     if(CNT <500)
278.                                         begin
279.                                                         display_2 <= clock_sec_H;
280.                                                         CNT <= CNT +1;
281.                                         end
282.
283.                                         else if(CNT >=500 && CNT<1000)
284.                                         begin
285.                                                         CNT <= CNT +1;
286.                                                         display_2 <= 4'b1111;
287.                                         end
288.                                         else if( CNT == 1000)
289.                                                         CNT <= 0;
290.                         end
291.
292.                         else if(pos_set == 2'b10)
293.                         begin
294.                                         display_4 <=     clock_min_H;
295.                                         display_2 <=     clock_sec_H;
296.                                         display_1 <= clock_sec_L;
297.
298.                                         if(CNT <500)
299.                                         begin
300.                                                         display_3 <= clock_min_L;
301.                                                         CNT <= CNT +1;
```

```
                                        end
                                        else if(CNT >=500 && CNT<1000)
                                        begin
                                                    CNT <= CNT +1;
                                                    display_3 <= 4'b1111;
                                        end
                                        else if( CNT == 1000)
                                                    CNT <= 0;
                            end
                            else if(pos_set == 2'b11)
                            begin
                                        display_3 <= clock_min_L;
                                        display_2 <=     clock_sec_H;
                                        display_1 <= clock_sec_L;
                                        if(CNT <500)
                                        begin
                                                    display_4 <= clock_min_H;
                                                    CNT <= CNT +1;
                                        end
                                        else if(CNT >=500 && CNT<1000)
                                        begin
                                                    CNT <= CNT +1;
                                                    display_4 <= 4'b1111;
                                        end
                                        else if( CNT == 1000)
                                                    CNT <= 0;
                            end
                        case(loop_time_set)
                            2'b00:
                            begin
                                            case(display_1)
                                                        4'b0000 : display_seg_1 = 7'b
    1000000;
                                                        4'b0001 : display_seg_1 = 7'b
    1111001;
                                                        4'b0010 : display_seg_1 = 7'b
    0100100;
                                                        4'b0011 : display_seg_1 = 7'b
    0110000;
                                                        4'b0100 : display_seg_1 = 7'b
    0011001;
                                                        4'b0101 : display_seg_1 = 7'b
    0010010;
                                                        4'b0110 : display_seg_1 = 7'b
    0000010;
                                                        4'b0111 : display_seg_1 = 7'b
    1111000;
                                                        4'b1000 : display_seg_1 = 7'b
    0000000;
                                                        4'b1001 : display_seg_1 = 7'b
    0010000;
```

```
                                                default   : display_seg_1 = 7'b1111111;
                                            endcase
                            end
                            2'b01:
                            begin
                                        case(display_2)
                                                4'b0000 : display_seg_2 = 7'b1000000;
                                                4'b0001 : display_seg_2 = 7'b1111001;
                                                4'b0010 : display_seg_2 = 7'b0100100;
                                                4'b0011 : display_seg_2 = 7'b0110000;
                                                4'b0100 : display_seg_2 = 7'b0011001;
                                                4'b0101 : display_seg_2 = 7'b0010010;
                                                4'b0110 : display_seg_2 = 7'b0000010;
                                                4'b0111 : display_seg_2 = 7'b1111000;
                                                4'b1000 : display_seg_2 = 7'b0000000;
                                                4'b1001 : display_seg_2 = 7'b0010000;
                                                default : display_seg_2 = 7'b1111111;
                                            endcase
                            end
                            2'b10:
                            begin
                                        case(display_3)
                                                4'b0000 : display_seg_3 = 7'b1000000;
                                                4'b0001 : display_seg_3 = 7'b1111001;
                                                4'b0010 : display_seg_3 = 7'b0100100;
                                                4'b0011 : display_seg_3 = 7'b0110000;
                                                4'b0100 : display_seg_3 = 7'b0011001;
                                                4'b0101 : display_seg_3 = 7'b0010010;
                                                4'b0110 : display_seg_3 = 7'b0000010;
                                                4'b0111 : display_seg_3 = 7'b1111000;
                                                4'b1000 : display_seg_3 = 7'b0000000;
                                                4'b1001 : display_seg_3 = 7'b0010000;
                                                default : display_seg_3 = 7'b1111111;
                                            endcase
                            end
                            2'b11:
                            begin
                                        case(display_4)
```

```
                                                    4'b0000 : display_seg_4 = 7'b1000000;
                                                    4'b0001 : display_seg_4 = 7'b1111001;
                                                    4'b0010 : display_seg_4 = 7'b0100100;
                                                    4'b0011 : display_seg_4 = 7'b0110000;
                                                    4'b0100 : display_seg_4 = 7'b0011001;
                                                    4'b0101 : display_seg_4 = 7'b0010010;
                                                    4'b0110 : display_seg_4 = 7'b0000010;
                                                    4'b0111 : display_seg_4 = 7'b1111000;
                                                    4'b1000 : display_seg_4 = 7'b0000000;
                                                    4'b1001 : display_seg_4 = 7'b0010000;
                                                    default : display_seg_4 = 7'b1111111;
                                                endcase
                                    end

                                endcase

                        end
end

end


endmodule
                        //ms
                    /*
                        case(loop)
                        2'b00:
                        begin
                                    case(display_1)
                                                4'b0000 : display_seg_1 = 7'b1000000;
                                                4'b0001 : display_seg_1 = 7'b1111001;
                                                4'b0010 : display_seg_1 = 7'b0100100;
                                                4'b0011 : display_seg_1 = 7'b0110000;
                                                4'b0100 : display_seg_1 = 7'b0011001;
                                                4'b0101 : display_seg_1 = 7'b0010010;
                                                4'b0110 : display_seg_1 = 7'b0000010;
                                                4'b0111 : display_seg_1 = 7'b1111000;
                                                4'b1000 : display_seg_1 = 7'b0000000;
                                                4'b1001 : display_seg_1 = 7'b0010000;
                                                default   : display_seg_1 = 7'b1111111;

                                        endcase
                        end
                        2'b01:
                        begin
                                    case(display_2)
                                                4'b0000 : display_seg_2 = 7'b1000000;
                                                4'b0001 : display_seg_2 = 7'b1111001;
                                                4'b0010 : display_seg_2 = 7'b0100100;
                                                4'b0011 : display_seg_2 = 7'b0110000;
```

```
                                    4'b0100 : display_seg_2 = 7'b0011001;
                                    4'b0101 : display_seg_2 = 7'b0010010;
                                    4'b0110 : display_seg_2 = 7'b0000010;
                                    4'b0111 : display_seg_2 = 7'b1111000;
                                    4'b1000 : display_seg_2 = 7'b0000000;
                                    4'b1001 : display_seg_2 = 7'b0010000;
                                    default : display_seg_2 = 7'b1111111;
                                endcase
            end
            2'b10:
            begin
                            case(display_3)
                                    4'b0000 : display_seg_3 = 7'b1000000;
                                    4'b0001 : display_seg_3 = 7'b1111001;
                                    4'b0010 : display_seg_3 = 7'b0100100;
                                    4'b0011 : display_seg_3 = 7'b0110000;
                                    4'b0100 : display_seg_3 = 7'b0011001;
                                    4'b0101 : display_seg_3 = 7'b0010010;
                                    4'b0110 : display_seg_3 = 7'b0000010;
                                    4'b0111 : display_seg_3 = 7'b1111000;
                                    4'b1000 : display_seg_3 = 7'b0000000;
                                    4'b1001 : display_seg_3 = 7'b0010000;
                                    default : display_seg_3 = 7'b1111111;
                                endcase
            end
            2'b11:
            begin
                            case(display_4)
                                    4'b0000 : display_seg_4 = 7'b1000000;
                                    4'b0001 : display_seg_4 = 7'b1111001;
                                    4'b0010 : display_seg_4 = 7'b0100100;
                                    4'b0011 : display_seg_4 = 7'b0110000;
                                    4'b0100 : display_seg_4 = 7'b0011001;
                                    4'b0101 : display_seg_4 = 7'b0010010;
                                    4'b0110 : display_seg_4 = 7'b0000010;
                                    4'b0111 : display_seg_4 = 7'b1111000;
                                    4'b1000 : display_seg_4 = 7'b0000000;
                                    4'b1001 : display_seg_4 = 7'b0010000;
                                    default : display_seg_4 = 7'b1111111;
                                endcase
            end

        endcase

        loop <= loop + 2'b01;
        if(loop == 2'b11)
        begin
                    loop <= 2'b00;
        end
        //
        //if(loop == pos_set)
        //begin


        //end
*/
```

key_filter.v

```
/*
        Student Name: Yang Li
        Studen Number: 1715977
*/
module key_filter(  clk,
                                                    system_reset,
                                                    key_in,
                                                    key_out

 );

 input                  clk;

 input                  system_reset,
                                    key_in;

output              key_out;

wire                        clk,
                                    system_reset,
                                    key_in;

reg                         key_out;


//    localparam TIME_20MS = 1_000_000;
    localparam TIME_20MS = 1000_000; //1000_000

    reg key_cnt;
    reg [20:0] cnt;

   always @(posedge clk or negedge system_reset) begin
        if(system_reset == 0)
            key_cnt <= 0;
        else if(cnt == TIME_20MS - 1)
            key_cnt <= 0;
        else if(key_cnt == 0 && key_out != key_in)
            key_cnt <= 1;
    end

    always @(posedge clk or negedge system_reset) begin
        if(system_reset == 0)
            cnt <= 0;
        else if(key_cnt) begin
            if(key_out == key_in)
                cnt <= 0;
            else
                cnt <= cnt + 1'b1;
        end
        else
            cnt <= 0;
    end
```

```
54.
55.      always @(posedge clk or negedge system_reset) begin
56.              if(system_reset == 0)
57.                  key_out <= 0;
58.              else if(cnt == TIME_20MS - 1)
59.                  key_out <= key_in;
60.      end
61. endmodule
62.
63.
64.
65.
66.
67. /*module key_filter(clk,
68.                                                      system_reset,
69.                                                      key,
70.                                                      key_out

71. );
72.
73. input                        clk;
74. input                    system_reset,
75.                                    key;
76.
77. output              key_out;
78.
79. reg                               key_out,
80.                                    key_value,
81.                                    key_flag,
82.                                    key_reg;
83.
84. reg[19:0]   delay_cnt;
85. //延时计数器
86. always@(posedge clk or negedge system_reset)
87. begin
88.              if(~system_reset)
89.              begin
90.                          key_reg <= 1'b1;
91.                          delay_cnt <= 20'b0;
92.              end
93.
94.              else
95.              begin
96.                           key_reg <= key;
97.                           if(key != key_reg)
98.                                    delay_cnt <= 20'd1000000;
99.                          else
100.                          begin
101.                                   if(delay_cnt > 20'd0)
102.                                            delay_cnt <= delay_cnt -
    1'b1;
103.                                   else
104.                                            delay_cnt <= 20'd0;
105.                          end
106.             end
107.end
108.
109.always@(posedge clk or negedge system_reset)
110.begin
111.             if(~system_reset)
112.             begin
```

```
                        key_value <= 1'b1;
                        key_flag <= 1'b0;
            end
            else
            begin
                        if(delay_cnt == 20'd1)
                        begin
                                    key_flag <= 1'b1;
                                    key_value <= key;
                        end

                        else
                        begin
                                    key_flag <= 1'b0;
                                    key_value <= key_value;
                        end
            end
end
always@(posedge clk)
begin
            if(key_flag && (~key_value))
                        key_out <= 1'b0;
            else
                        key_out <= 1'b1;
end
endmodule

*/
```