

---

# 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

|   |   |
|---|---|
| Abstract .....                          | 1 |
| Design Objectives .....                 | 3 |
| Function Description.....               | 3 |
| System overview with a diagram .....    | 4 |
| Sythesised circuits.....                | 4 |
| a. Frequency Divider Module .....       | 4 |
| b. Digital Clock Module .....           | 5 |
| (1) Digital Clock Timing Function ..... | 6 |
| (2) Time Setting Function .....         | 6 |
| c. Stopwatch Module .....               | 6 |
| d. Key Filter Module.....               | 7 |

|   |    |
|---|----|
| e. Decoding Scan Display Module .....   | 8  |
| (1) Display function .....              | 8  |
| (2) Time Setting blinking function..... | 9  |
| f. Top Module .....                     | 9  |
| Simulation of the core components.....  | 10 |
| a. Frequency Division Function .....    | 10 |
| b. Digital Clock Timing Function.....   | 11 |
| c. Time Setting Function .....          | 11 |
| d. Stopwatch Function .....             | 12 |
| e. Decoding Scan Display Function ..... | 12 |
| f. Key Filter Function.....             | 13 |
| Conclusions.....                        | 13 |
| Appendix.....                           | 14 |

## 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

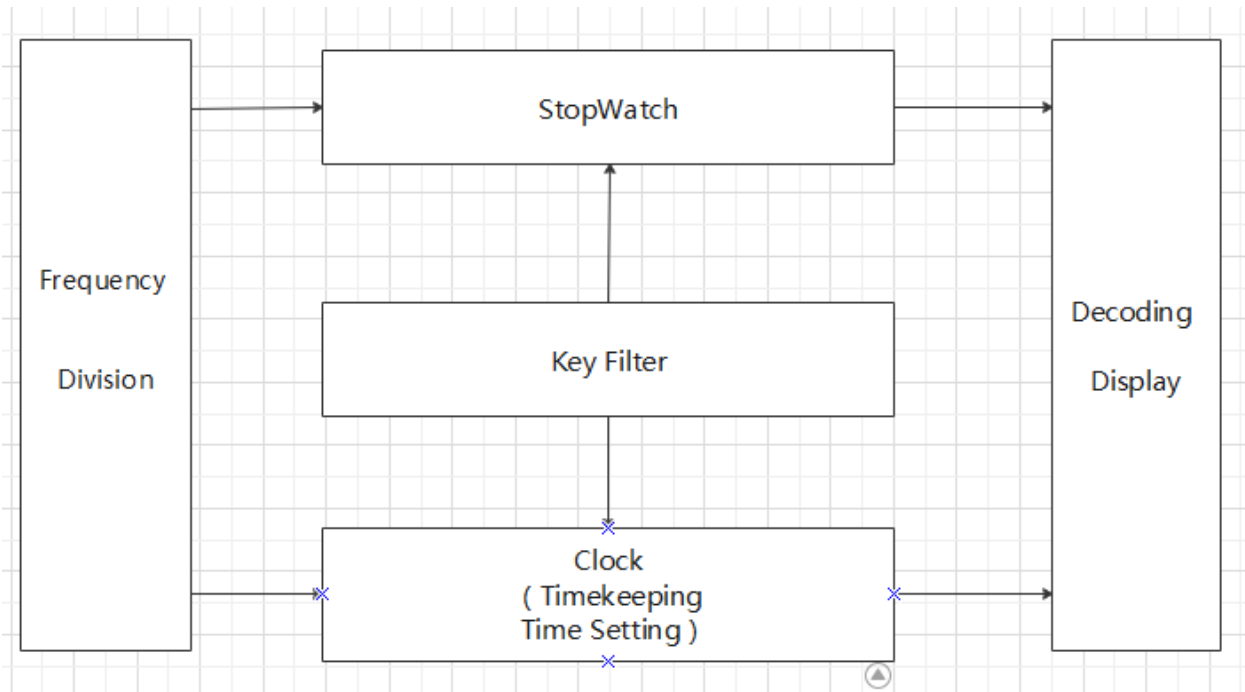


Diagram 1: Schematic Diagram of Digital Clock

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.

## Synthesised 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 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

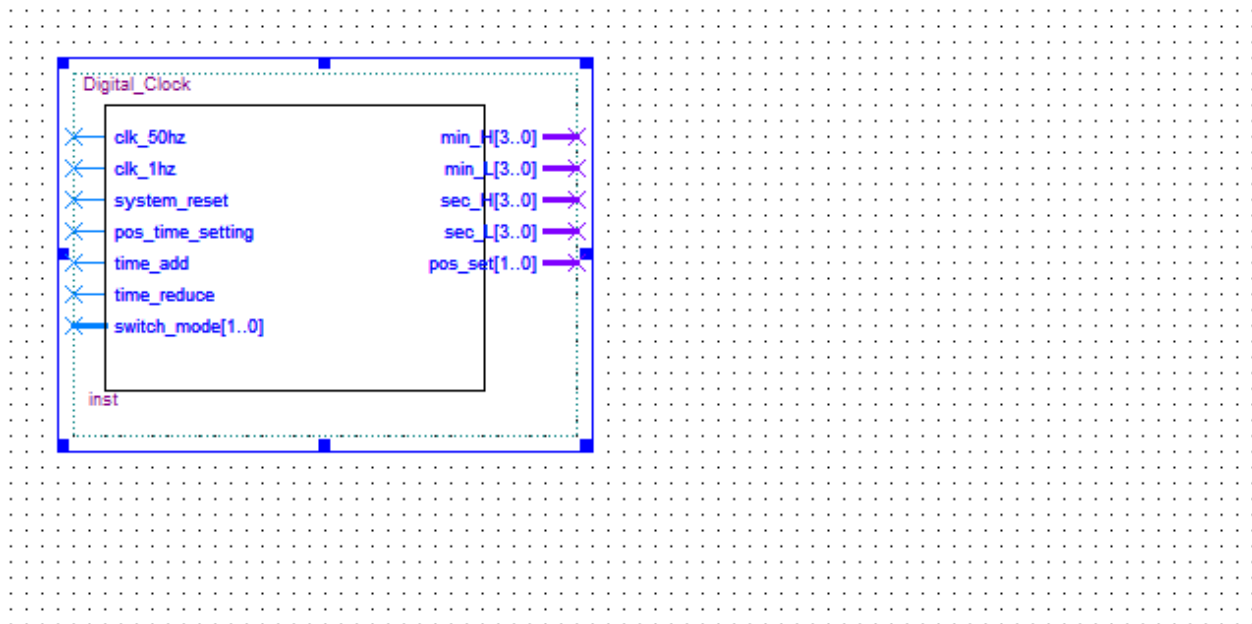


Figure 3: The Synthesised Circuit of Digital Clock Module

## (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

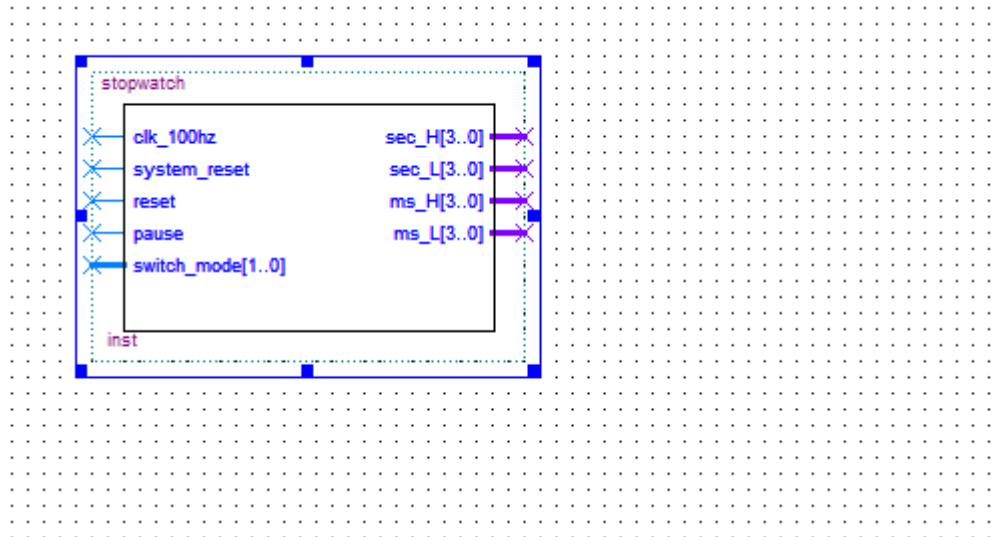


Figure 4: The Synthesised Circuit of Stopwatch Module

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.  $ms\_L \leq ms\_L$ .

Reset:

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

#### d. Key Filter Module

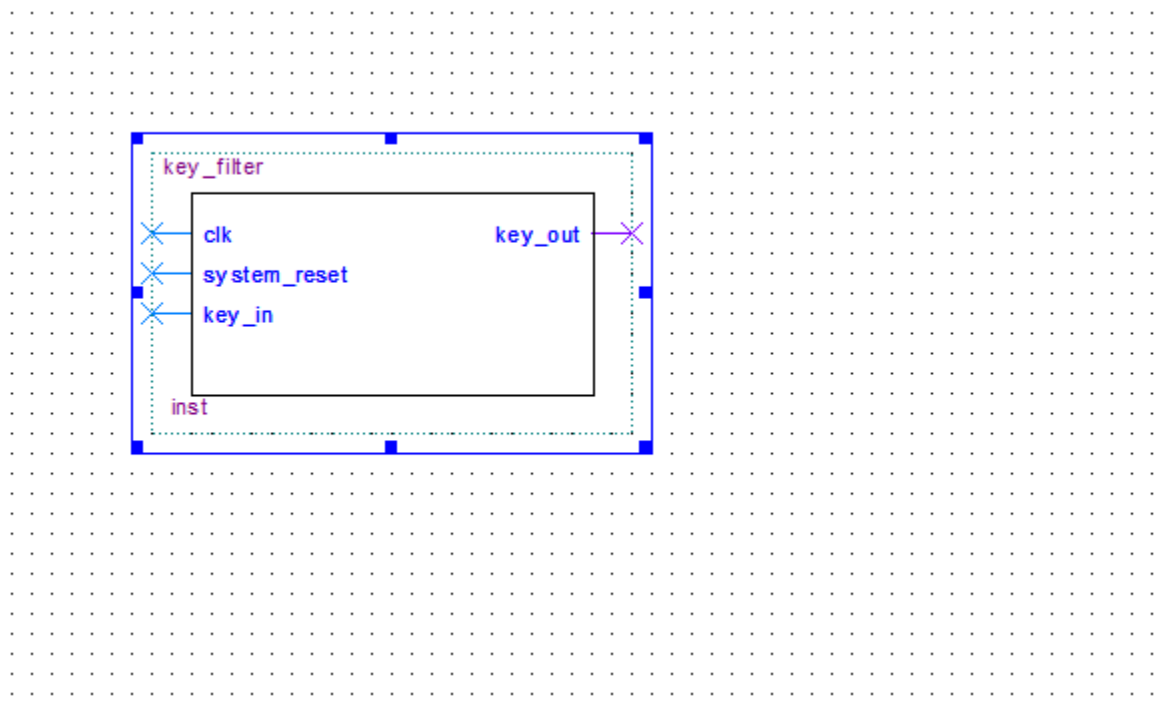


Figure 5: The Synthesised Circuit of Key Filter Module

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

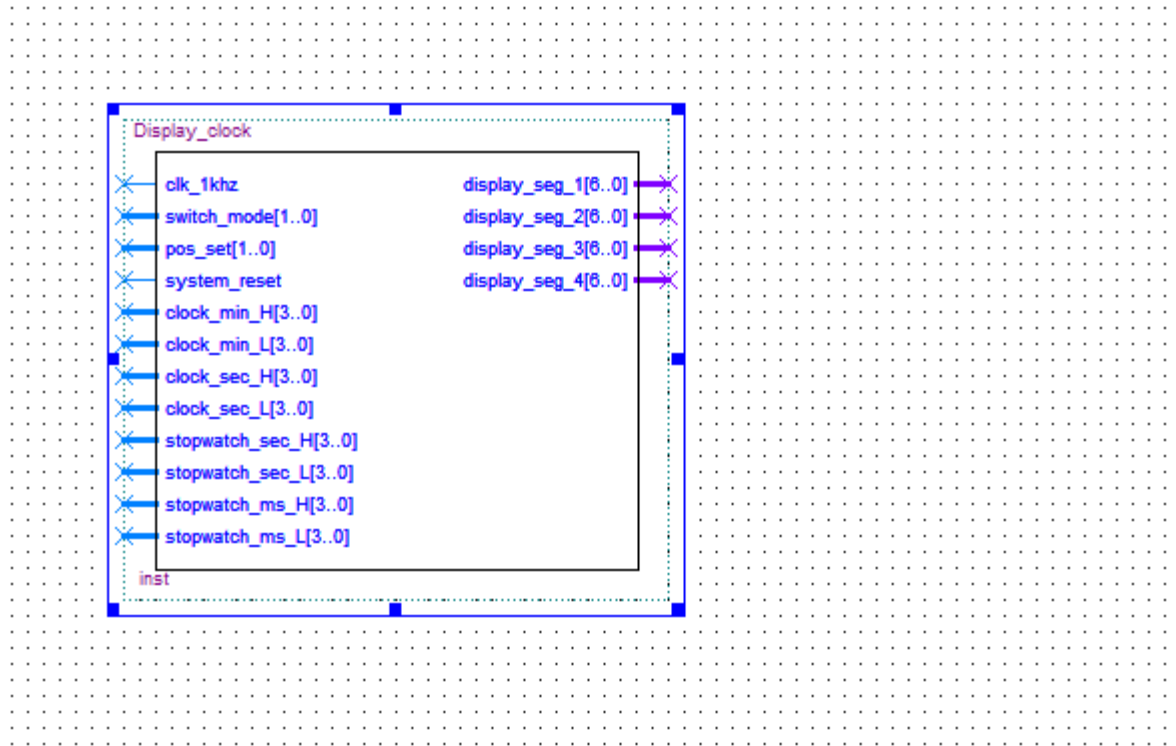


Figure 6: The Synthesised Circuit of Decoding Scan Display Module

##### (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.

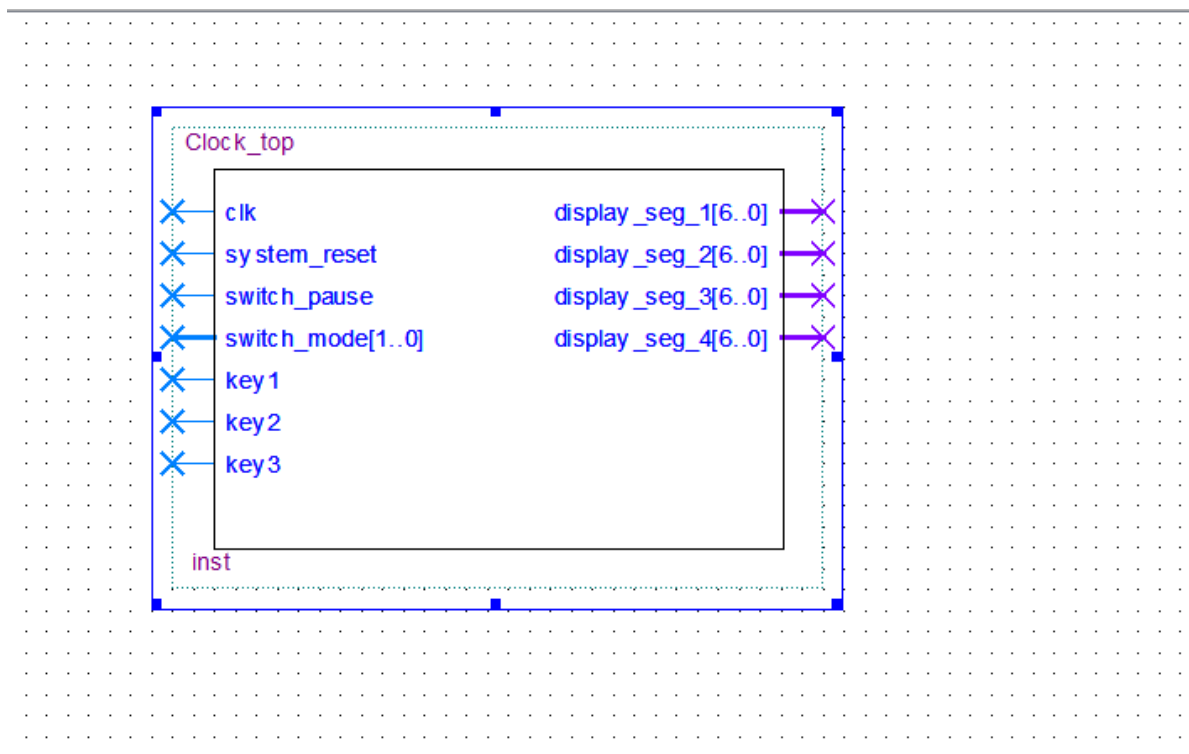


Figure 7: The Sythesised Circuit of Top Module

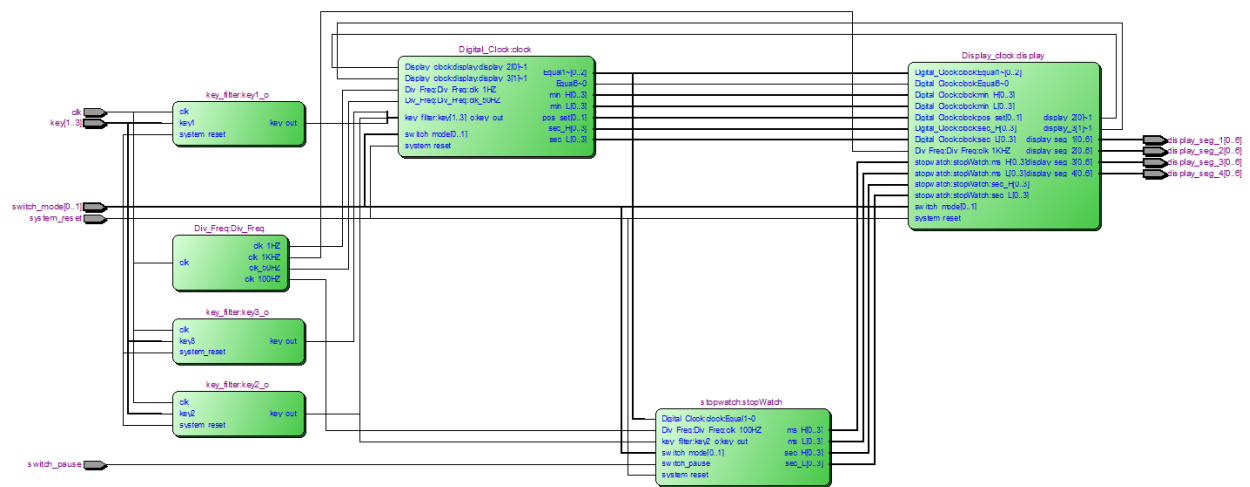


Figure 8: The Circuit of Digital Clock

## Simulation of the core components

### a. Frequency Division Function

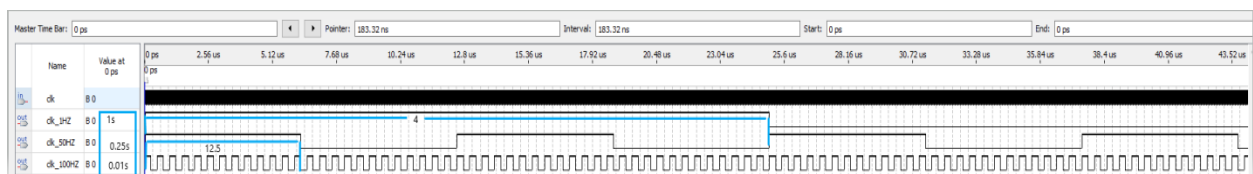


Figure 9: Simulation results of Frequency Division Function

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

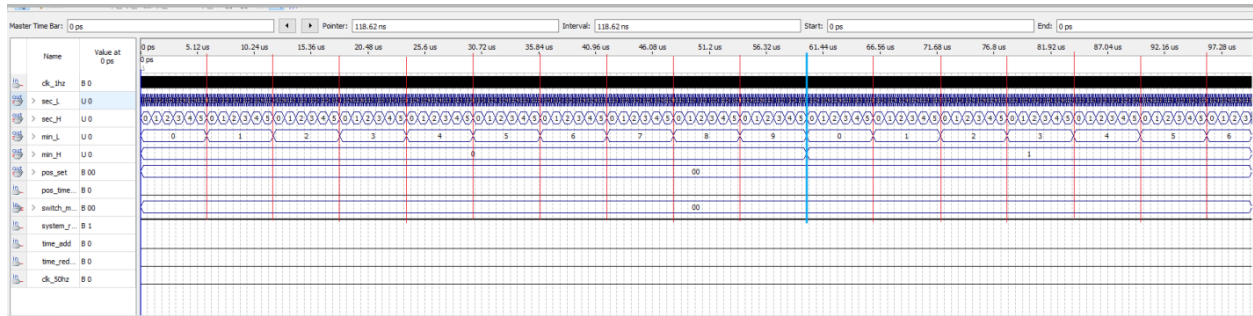


Figure 10: Simulation results of Digital Clock Timing Function

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

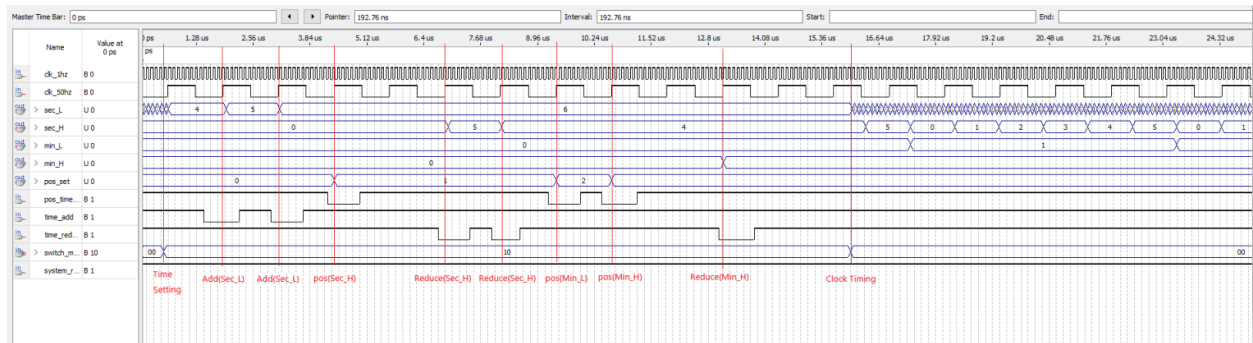


Figure 11: Simulation results of Timing Setting Function

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.

Timing diagram showing the sequence of events for the 'pause' and 'Reset' signals. The diagram includes a Master Time Bar at the top with a pointer at 21.35 us and an interval of 21.35 us. The signals shown are: ck\_100Hz (clock), ms\_H (high), ms\_L (low), pause (low), reset (low), sec\_H (high), sec\_L (low), switch\_m (high), and system\_r (low). The 'pause' signal is shown as a low pulse, and the 'Reset' signal is shown as a low pulse. The diagram is annotated with red arrows and text: 'pause' and 'Reset'.

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.

The timing diagram illustrates the relationship between the system\_reset signal and the clock timing. The top section, labeled "Clock Timing", shows the clock signal (clk\_3Hz) and the system\_reset signal (B0) with various timing parameters such as 0 ps, 5.12 us, 10.24 us, 15.36 us, 20.48 us, 25.6 us, 30.72 us, 35.84 us, 40.96 us, 46.08 us, 51.2 us, 56.32 us, 61.44 us, 66.56 us, 71.68 us, 76.8 us, 81.92 us, 87.04 us, 92.16 us, and 97.28 us. The bottom section, labeled "Time Setting", shows the system\_reset signal (B1) with a value of 01. A red arrow points from the "Time Setting" section to the "Clock Timing" section, indicating the transition from time setting to clock timing.

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

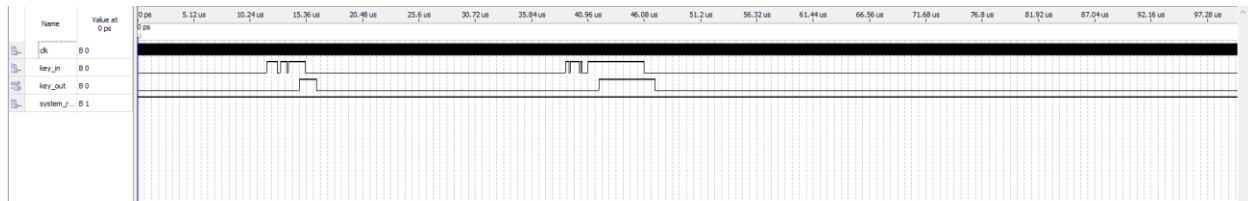


Figure 14: Simulation results of Key Filter Function

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

```
1.  /*
2.      Student Name: Yang Li
3.      Studen Number: 1715977
4.
5.      SW9 DOWN ---- System Reset
6.      SW9 UP---- System Start
7.
8.      Digital Clock | SW9 UP | Time Set | Stopwatch |
9.      SW9 UP | SW0 DOWN | | |
10.      | SW0 UP | SW0 DOWN | | |
11.      | SW1 DOWN | SW1 DOWN | | |
12.      | SW1 UP | | | |
13.      - | | | Function :
14.      | | | Function:
15.      | SW2 UP(Pause) | |
16.      | KEY1(Position) | |
17.
18.      KEY2 (Reset)
19.      KEY2(Add Time
20.
21.      - | KEY3(Reduce Time) |
22.
23.
24.
25.
26.
27.
28.
29.  */

18. module Clock_top(clk,
19.
20.      system_reset,
21.      switch_pause,
22.      switch_mode,
23.      key1,
24.      key2,
25.      key3,
26.      display_seg_1,
27.      display_seg_2,
28.      display_seg_3,
29.      display_seg_4,
```

```

30. );
31.
32. input clk;
33. input system_reset,
34.       switch_pause,
35.       key1,
36.       key2,
37.       key3;
38. input[1:0] switch_mode;
39. //display
40. output[6:0] display_seg_1,
41.            display_seg_2,
42.            display_seg_3,
43.            display_seg_4;
44. //clock frequency
45. wire clk_1HZ,
46.       clk_50HZ,
47.       clk_100HZ,
48.       clk_1KHZ;
49.
50. wire[1:0] pos_set;
51.
52.
53. //key output
54. wire key1_out,
55.       key2_out,
56.       key3_out;
57. //clock_time
58. wire[3:0] clock_min_H,
59.           clock_min_L,
60.           clock_sec_H,
61.           clock_sec_L;
62. //stopwatch
63. wire[3:0] stopwatch_sec_H,
64.           stopwatch_sec_L,
65.           stopwatch_ms_H,
66.           stopwatch_ms_L;
67.
68.
69.
70. key_filter key1_o( .clk (c
71.                   (system_reset), .system_reset
72.                   (key1), .key_in
73.                   (key1_out) .key_out
74. );
75.
76. key_filter key2_o( .clk (c
77.                   (system_reset), .system_reset
78.                   (key2), .key_in
79.                   (key2_out) .key_out
80. );
81.

```

|      |                            |               |               |
|------|----------------------------|---------------|---------------|
| 82.  | key_filter key3_o(         | .clk          | (c            |
|      | 1k),                       |               |               |
| 83.  |                            |               | .system_reset |
|      | (system_reset),            |               |               |
| 84.  |                            |               | .key_in       |
|      | (key3),                    |               |               |
| 85.  |                            |               | .key_out      |
|      | (key3_out)                 |               |               |
| 86.  | );                         |               |               |
| 87.  |                            |               |               |
| 88.  | Div_Freq Div_Freq(         |               |               |
| 89.  |                            | .clk          |               |
|      | (clk),                     |               |               |
| 90.  |                            | .clk_1HZ      |               |
|      | (clk_1HZ),                 |               |               |
| 91.  |                            | .clk_50HZ     |               |
|      | (clk_50HZ),                |               |               |
| 92.  |                            | .clk_100HZ    |               |
|      | (clk_100HZ),               |               |               |
| 93.  |                            | .clk_1KHZ     |               |
|      | (clk_1KHZ)                 |               |               |
| 94.  |                            |               | );            |
| 95.  | stopwatch stopWatch(       |               |               |
| 96.  |                            | .clk_100hz    |               |
|      | (clk_100HZ),               |               |               |
| 97.  |                            | .system_reset |               |
|      | (system_reset), //switch9  |               |               |
| 98.  |                            | .reset        |               |
|      | (key2_out), //key2         |               |               |
| 99.  |                            | .pause        |               |
|      | (switch_pause), //switch 2 |               |               |
| 100. |                            | .switch_mode  |               |
|      | (switch_mode),             |               |               |
| 101. |                            | .sec_H        |               |
|      | (stopwatch_sec_H),         |               |               |
| 102. |                            | .sec_L        |               |
|      | (stopwatch_sec_L),         |               |               |
| 103. |                            | .ms_H         |               |
|      | (stopwatch_ms_H),          |               |               |
| 104. |                            | .ms_L         |               |
|      | (stopwatch_ms_L)           |               |               |
| 105. |                            |               | );            |
| 106. |                            |               |               |
| 107. |                            |               |               |
| 108. | Display_clock display (    |               |               |
| 109. |                            | .clk_1khz     |               |
|      | (clk_1KHZ),                |               |               |
| 110. |                            | .switch_mode  |               |
|      | (switch_mode),             |               |               |
| 111. |                            | .pos_set      |               |
|      | (pos_set),                 |               |               |
| 112. |                            | .system_reset |               |
|      | (system_reset),            |               |               |
| 113. |                            | .clock_min_H  |               |
|      | (clock_min_H),             |               |               |
| 114. |                            | .clock_min_L  |               |
|      | (clock_min_L),             |               |               |
| 115. |                            | .clock_sec_H  |               |
|      | (clock_sec_H),             |               |               |



```

116.                                     .clock_sec_L
117.                                     .stopwatch_sec_H
118.         (stopwatch_sec_H),
119.         (stopwatch_sec_L),
120.         stopwatch_ms_H),
121.         (stopwatch_ms_L),
122.                                     .display_seg_1
123.         (display_seg_1),
124.         (display_seg_2),
125.         (display_seg_3),
126.         (display_seg_4)
127.                                     .display_seg_2
128.                                     .display_seg_3
129.                                     .display_seg_4
130.                                     );
131.
132. Digital_Clock clock (
133.         (clk_50HZ),
134.         (clk_1HZ),
135.         (system_reset),
136.         (key1_out), //key1
137.         (key2_out), //key2
138.         (key3_out), //key3
139.         (switch_mode),
140.         (clock_min_H),
141.         (clock_min_L),
142.         (clock_sec_H),
143.         (clock_sec_L),
144.         //modify
145.         (pos_set)
146.         //over
147. );
148.
149. endmodule

```

```

1.  /*
2.      Student Name: Yang Li
3.      Studen Number: 1715977
4.  */
5.  module Div_Freq (clk,
6.                  clk_1HZ,
7.                  clk_50HZ,
8.                  clk_100HZ,
9.                  clk_1KHZ);
10. input            clk;
11. output           clk_1HZ,
12.             clk_50HZ,
13.             clk_100HZ,
14.             clk_1KHZ;
15. reg              clk_1HZ,
16.             clk_50HZ,
17.             clk_100HZ,
18.             clk_1KHZ;
19. integer           count1 = 0,
20.             count2 = 0,
21.             count100 = 0,
22.             count1k = 0;
23.
24.
25. always@(posedge clk)
26. begin
27.
28.     count1<=count1+1;
29.     if(count1<25_000_000)           //25_000_000
30.         clk_1HZ<=1;
31.     else if(count1<49_999_999)      //49999999
32.         clk_1HZ<=0;
33.     if(count1==49_999_999)          //49999999
34.         count1<=0;
35. end
36. always@(posedge clk)
37. begin//0.4s
38.     count2<=count2+1;
39.     if(count2<625)                 //1250000      10_000_000(0.4s)
40.         clk_50HZ<=1;
41.     else if(count2<12_49)          //2499999      19_999_999
42.         clk_50HZ<=0;
43.     if(count2==12_49)              //2499999      19_999_999
44.         count2<=0;
45. end
46. always@(posedge clk)
47. begin
48.     count100<=count100+1;
49.
50.     if(count100<25)//250000
51.         clk_100HZ<=1;
52.     else if(count100<49)//499999
53.         clk_100HZ<=0;
54.     if(count100==49)//499999
55.         count100<=0;
56. end
57.
58.
59. always@(posedge clk)
60. begin
61.     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.
7.                      system_reset,
8.                      reset, //key2
9.                      pause, //switch sw2
10.                     switch_mode,
11.                     sec_H,
12.                     sec_L,
13.                     ms_H,
14.                     ms_L);
15.
16. input                clk_100hz;
17.
18. input                reset,
19.                     pause,
20.                     system_reset;
21. input [1:0 ]         switch_mode;
22. output [3:0]         sec_H,
23.                     sec_L,
24.                     ms_H,
25.                     ms_L;
26.
27. reg [3:0]            sec_H,
28.                     sec_L,
29.                     ms_H,
30.                     ms_L;
31.
32. always @(posedge clk_100hz )
33. begin
34.     if(system_reset ==0)
35.     begin
36.         ms_L <= 4'b0000;
37.         ms_H <= 4'b0000;
38.         sec_H <= 4'b0000;
39.         sec_L <= 4'b0000;
40.     end
41.     else
42.     begin
43.         if(switch_mode == 2'b01)
44.         begin
45.             if(~reset)
46.             begin
47.                 ms_L <= 4'b0000;

```

```

47.          ms_H <= 4'b0000;
48.          sec_H <= 4'b0000;
49.          sec_L <= 4'b0000;
50.      end
51.
52.      else if(pause)
53.      begin
54.          ms_L <= ms_L;
55.          ms_H <= ms_H;
56.          sec_H <= sec_H;
57.          sec_L <= sec_L;
58.      end
59.
60.      else
61.      begin
62.          ms_L <= ms_L + 4'b0001;
63.          if(ms_L == 4'b1001)
64.          begin
65.              ms_L <= 4'b0000;
66.              ms_H <= ms_H + 4'b0
001;
67.          end
68.
69.          if(ms_L == 4'b1001 && ms_H ==
4'b1001)
70.          begin
71.              ms_H <= 4'b000;
72.              sec_L <= sec_L + 4
'b0001;
73.          end
74.
75.          if(sec_L == 4'b1001 && (ms_H ==
4'b0101 && ms_L == 4'b1001))
76.          begin
77.              sec_L <= 4'b0000;
78.              sec_H <= sec_H + 4
'b0001;
79.          end
80.
81.          if( (sec_H == 4'b0101 && sec_
L == 4'b1001) && (ms_H == 4'b0101 && ms_L == 4'b1001) )
82.          begin
83.              ms_L <= 4'b0000;
84.              ms_H <= 4'b0000;
85.              sec_H <= 4'b0000;
86.              sec_L <= 4'b0000;
87.          end
88.      end
89.  end
90. end
91. end
92.
93. endmodule

```

## Digital\_Clock.v

```
1.  /*
2.      Student Name: Yang Li
3.      Studen Number: 1715977
4.  */
5.  module Digital_Clock(clk_50hz,
6.
7.                                clk_1hz,
8.                                system_reset,
9.                                pos_time_setting,//key
10.
11.                                1
12.                                //key2
13.                                time_add,
14.                                //key3
15.                                time_reduce,
16.                                switch_mode,
17.                                min_H,
18.                                min_L,
19.                                sec_H,
20.                                sec_L,
21.                                //modify
22.                                pos_set);
23.                                //over
24.
25.  input                        clk_50hz,
26.                                clk_1hz;
27.
28.  input                        system_reset;
29.
30.  input                        pos_time_setting,
31.                                time_add,
32.                                time_reduce;
33.
34.  input [1:0]                  switch_mode;
35.
36.  output [3:0]                  min_H,
37.                                min_L,
38.                                sec_H,
39.                                sec_L;
40.
41.  reg [3:0]                     min_H,
42.                                min_L,
43.                                sec_H,
44.                                sec_L;
45.
46.  reg[3:0]                      min_H_set,
47.                                min_L_set,
48.                                sec_H_set,
49.                                sec_L_set;
50.
51.  reg [3:0]                     temp_min_H,
52.                                temp_min_L,
53.                                temp_sec_H,
54.                                temp_sec_L;
55.  //modify
56.  output[1:0]                   pos_set;
57.  //over
```

```

53. reg[1:0]                                pos_set;
54.
55. reg                                     loop;
56. reg[3:0]                               conflict;
57.
58. //clock count
59. always@(posedge clk_1hz)
60. begin
61.     if(system_reset == 0)
62.     begin
63.         min_H <= 4'b0000;
64.         min_L <= 4'b0000;
65.         sec_H <= 4'b0000;
66.         sec_L <= 4'b0000;
67.         temp_sec_H <= 4'b0000;
68.         temp_sec_L <= 4'b0000;
69.         temp_min_H <= 4'b0000;
70.         temp_min_L <= 4'b0000;
71.
72.     end
73.     else begin
74.         if(switch_mode == 2'b10)
75.         begin
76.             sec_H <= sec_H_set;
77.             sec_L <= sec_L_set;
78.             min_H <= min_H_set;
79.             min_L <= min_L_set;
80.         end
81.
82.         //else if(switch_mode == 2'b00)
83.         else if(switch_mode == 2'b00)
84.         begin
85.             sec_L <= sec_L + 4'b0001;
86.
87.             if(sec_L == 4'b1001)
88.             begin
89.                 sec_L <= 4'b0000;
90.                 sec_H <= sec_H + 4'b0001;
91.             end
92.
93.             if(sec_H == 4'b0101 && sec_L == 4'b1001)
94.             begin
95.                 sec_H <= 4'b0000;
96.                 min_L <= min_L + 4'b0001;
97.             end
98.
99.             if(min_L == 4'b1001 && (sec_H == 4'b0101 && sec_L == 4'
100. b1001))
101.             begin
102.                 min_L <= 4'b0000;
103.                 min_H <= min_H + 4'b0001;
104.             end
105.             if((min_H >= 4'b0101 && min_L >= 4'b1001) && (sec_H >=
106. 4'b0101 && sec_L >= 4'b1001))
107.             begin
108.                 min_H <= 4'b0000;
109.                 min_L <= 4'b0000;
110.                 sec_H <= 4'b0000;
111.                 sec_L <= 4'b0000;
112.             end
113.         end
114.     end
115. end

```

```

112.                temp_sec_H <=      sec_H;
113.                temp_sec_L <=      sec_L;
114.                temp_min_H <=      min_H;
115.                temp_min_L <=      min_L;
116.
117.
118.                end
119.            end
120.        end
121.    // _____
122.
123.    // _____
124.    //time_setting
125.    always@(posedge clk_50hz )//posedge clk_50hz )//or posedge pos_time_setting o
        r )
126.    begin
127.        if(system_reset == 0)
128.        begin
129.            sec_H_set <= 4'b0000;
130.            sec_L_set  <= 4'b0000;
131.            min_H_set <= 4'b0000;
132.            min_L_set <= 4'b0000 ;
133.            pos_set   <= 2'b00;
134.
135.
136.            end
137.
138.        else
139.        begin
140.
141.            if(switch_mode == 2'b10)
142.            begin
143.                if(loop == 0)
144.                begin
145.                    sec_H_set <= temp_sec_H;
146.                    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 )
147.                    min_H_set <= temp_min_H;
148.                    min_L_set <= temp_min_L ;
149.                    loop <= loop +1;
150.
151.                end
152.                // determine the position which is changed
153.
154.
155.                if(~pos_time_setting)
156.
157.                begin
158.                    if(pos_set == 2'b11)
159.                    begin
160.                        pos_set <= 2'b00;
161.                    end
162.                    else
163.                        pos_set <= pos_set+2'b01;
164.
165.                end
166.                // second_LOW - time setting
167.                if(pos_set==2'b00)
168.                begin
169.                    if(~time_add)

```

```

169.
170.             begin
171.                 if(sec_L_set == 4'b1001)
172.                     begin
173.                         sec_L_set <=
174.                             4'b0000;
175.                     end
176.                 else
177.                     sec_L_set <=
178.                         sec_L_set + 4'b0001;
179.                     end
180.                 if(~time_reduce)
181.                     begin
182.                         if(sec_L_set == 4'b0000)
183.                             begin
184.                                 sec_L_set <=
185.                                     4'b1001;
186.                             end
187.                         else
188.                             sec_L_set <=
189.                                 sec_L_set - 4'b0001;
190.                             end
191.                     end
192.                 //second_HIGH -time setting
193.                 if(pos_set==2'b01)
194.                     begin
195.                         if(~time_add)
196.                             begin
197.                                 if(sec_H_set == 4'b0101)
198.                                     begin
199.                                         sec_H_set <=
200.                                             4'b0000;
201.                                     end
202.                                 else
203.                                     sec_H_set <=
204.                                         sec_H_set + 4'b0001;
205.                                     end
206.                                 if(~time_reduce)
207.                                     begin
208.                                         if(sec_H_set == 4'b0000)
209.                                             begin
210.                                                 sec_H_set <=
211.                                                     4'b0101;
212.                                             end
213.                                         else
214.                                             sec_H_set <=
215.                                                 sec_H_set - 4'b0001;
216.                                             end
217.                                     end
218.                                 end
219.                             //min_LOW -time setting
220.                             if(pos_set==2'b10)

```



[illegible]



```

36.                                     stopwatch_ms_H,
37.                                     stopwatch_ms_L;
38.
39. output [6:0]                        display_seg_1,
40.                                     display_seg_2,
41.                                     display_seg_3,
42.                                     display_seg_4;
43.
44. reg [6:0]                            display_seg_1,
45.                                     display_seg_2,
46.                                     display_seg_3,
47.                                     display_seg_4;
48. reg [3:0]                            display_1,
49.                                     display_2,
50.                                     display_3,
51.                                     display_4;
52.
53. reg [1:0]                            loop_clock,
54.                                     loop_stopwatch,
55.                                     loop_time_set;
56. integer                              CNT;
57.
58. always@(posedge clk_1khz)
59. begin
60. if(system_reset ==0)
61. begin
62.         display_seg_1 = 7'b0000000;
63.         display_seg_2 = 7'b0000000;
64.         display_seg_3 = 7'b0000000;
65.         display_seg_4 = 7'b0000000;
66. end
67.
68. else
69. begin
70.         if(switch_mode == 2'b01)
71.         begin
72.                 display_4 <= stopwatch_sec_H;
73.                 display_3 <= stopwatch_sec_L;
74.                 display_2 <= stopwatch_ms_H;
75.                 display_1 <= stopwatch_ms_L;
76.
77.                 case(loop_stopwatch)
78.                     2'b00:
79.                     begin
80.                                     case(display_1)
81.                                     4'b0000 : display_
82.                                     seg_1 = 7'b1000000;
83.                                     4'b0001 : display_
84.                                     seg_1 = 7'b1111001;
85.                                     4'b0010 : display_
86.                                     seg_1 = 7'b0100100;
87.                                     4'b0011 : display_
88.                                     seg_1 = 7'b0110000;
89.                                     4'b0100 : display_
90.                                     seg_1 = 7'b0011001;
91.                                     4'b0101 : display_
92.                                     seg_1 = 7'b0010010;
93.                                     4'b0110 : display_
94.                                     seg_1 = 7'b0000010;
95.                                     4'b0111 : display_
96.                                     seg_1 = 7'b0000000;
97.                                     4'b1000 : display_
98.                                     seg_1 = 7'b0000000;
99.                                     4'b1001 : display_
100.                                    seg_1 = 7'b0000000;
101.                                    4'b1010 : display_
102.                                    seg_1 = 7'b0000000;
103.                                    4'b1011 : display_
104.                                    seg_1 = 7'b0000000;
105.                                    4'b1100 : display_
106.                                    seg_1 = 7'b0000000;
107.                                    4'b1101 : display_
108.                                    seg_1 = 7'b0000000;
109.                                    4'b1110 : display_
110.                                    seg_1 = 7'b0000000;
111.                                    4'b1111 : display_
112.                                    seg_1 = 7'b0000000;
113.                                end
114.                            end
115.                    end
116.                end
117.            end
118.        end
119.    end

```

```

88.                                     4'b0111 : display_
    seg_1 = 7'b1111000;
89.                                     4'b1000 : display_
    seg_1 = 7'b0000000;
90.                                     4'b1001 : display_
    seg_1 = 7'b0010000;
91.                                     default : displa
    y_seg_1 = 7'b1111111;
92.                                     endcase
93.                                     end
94.                                     2'b01:
95.                                     begin
96.                                     case(display_2)
97.                                     4'b0000 : display_
    seg_2 = 7'b1000000;
98.                                     4'b0001 : display_
    seg_2 = 7'b1111001;
99.                                     4'b0010 : display_
    seg_2 = 7'b0100100;
100.                                    4'b0011 : display
    _seg_2 = 7'b0110000;
101.                                    4'b0100 : display
    _seg_2 = 7'b0011001;
102.                                    4'b0101 : display
    _seg_2 = 7'b0010010;
103.                                    4'b0110 : display
    _seg_2 = 7'b0000010;
104.                                    4'b0111 : display
    _seg_2 = 7'b1111000;
105.                                    4'b1000 : display
    _seg_2 = 7'b0000000;
106.                                    4'b1001 : display
    _seg_2 = 7'b0010000;
107.                                    default : display
    _seg_2 = 7'b1111111;
108.                                    endcase
109.                                    end
110.                                    2'b10:
111.                                    begin
112.                                    case(display_3)
113.                                    4'b0000 : display
    _seg_3 = 7'b1000000;
114.                                    4'b0001 : display
    _seg_3 = 7'b1111001;
115.                                    4'b0010 : display
    _seg_3 = 7'b0100100;
116.                                    4'b0011 : display
    _seg_3 = 7'b0110000;
117.                                    4'b0100 : display
    _seg_3 = 7'b0011001;
118.                                    4'b0101 : display
    _seg_3 = 7'b0010010;
119.                                    4'b0110 : display
    _seg_3 = 7'b0000010;
120.                                    4'b0111 : display
    _seg_3 = 7'b1111000;
121.                                    4'b1000 : display
    _seg_3 = 7'b0000000;
122.                                    4'b1001 : display
    _seg_3 = 7'b0010000;

```

```

123.                                     default : display
    _seg_3 = 7'b1111111;
124.                                     endcase
125.                                     end
126.                                     2'b11:
127.                                     begin
128.                                     case(display_4)
129.                                     4'b0000 : display
    _seg_4 = 7'b1000000;
130.                                     4'b0001 : display
    _seg_4 = 7'b1111001;
131.                                     4'b0010 : display
    _seg_4 = 7'b0100100;
132.                                     4'b0011 : display
    _seg_4 = 7'b0110000;
133.                                     4'b0100 : display
    _seg_4 = 7'b0011001;
134.                                     4'b0101 : display
    _seg_4 = 7'b0010010;
135.                                     4'b0110 : display
    _seg_4 = 7'b0000010;
136.                                     4'b0111 : display
    _seg_4 = 7'b1111000;
137.                                     4'b1000 : display
    _seg_4 = 7'b0000000;
138.                                     4'b1001 : display
    _seg_4 = 7'b0010000;
139.                                     default : display
    _seg_4 = 7'b1111111;
140.                                     endcase
141.                                     end
142.
143.                                     endcase
144.
145.                                     loop_stopwatch <= loop_stopwatch + 2'b01;
146.                                     if(loop_stopwatch == 2'b11)
147.                                     begin
148.                                     loop_stopwatch <= 2'b00;
149.                                     end
150.                                     end
151.                                     //clock
152.                                     else if( switch_mode == 2'b00)
153.                                     begin
154.                                     display_4 <= clock_min_H;
155.                                     display_3 <= clock_min_L;
156.                                     display_2 <= clock_sec_H;
157.                                     display_1 <= clock_sec_L;
158.
159.                                     loop_clock <= loop_clock + 2'b01;
160.                                     if(loop_clock == 2'b11)
161.                                     begin
162.                                     loop_clock <= 2'b00;
163.                                     end
164.
165.                                     case(loop_clock)
166.                                     2'b00:
167.                                     begin
168.                                     case(display_1)
169.                                     4'b0000 : display
    _seg_1 = 7'b1000000;

```

```

170.                                     4'b0001 : display
    _seg_1 = 7'b1111001;
171.                                     4'b0010 : display
    _seg_1 = 7'b0100100;
172.                                     4'b0011 : display
    _seg_1 = 7'b0110000;
173.                                     4'b0100 : display
    _seg_1 = 7'b0011001;
174.                                     4'b0101 : display
    _seg_1 = 7'b0010010;
175.                                     4'b0110 : display
    _seg_1 = 7'b0000010;
176.                                     4'b0111 : display
    _seg_1 = 7'b1111000;
177.                                     4'b1000 : display
    _seg_1 = 7'b0000000;
178.                                     4'b1001 : display
    _seg_1 = 7'b0010000;
179.                                     default : displ
    ay_seg_1 = 7'b1111111;
180.                                     endcase
181.                                     end
182.                                2'b01:
183.                                begin
184.                                     case(display_2)
185.                                         4'b0000 : display
    _seg_2 = 7'b1000000;
186.                                         4'b0001 : display
    _seg_2 = 7'b1111001;
187.                                         4'b0010 : display
    _seg_2 = 7'b0100100;
188.                                         4'b0011 : display
    _seg_2 = 7'b0110000;
189.                                         4'b0100 : display
    _seg_2 = 7'b0011001;
190.                                         4'b0101 : display
    _seg_2 = 7'b0010010;
191.                                         4'b0110 : display
    _seg_2 = 7'b0000010;
192.                                         4'b0111 : display
    _seg_2 = 7'b1111000;
193.                                         4'b1000 : display
    _seg_2 = 7'b0000000;
194.                                         4'b1001 : display
    _seg_2 = 7'b0010000;
195.                                         default : display
    _seg_2 = 7'b1111111;
196.                                     endcase
197.                                     end
198.                                2'b10:
199.                                begin
200.                                     case(display_3)
201.                                         4'b0000 : display_seg
    _3 = 7'b1000000;
202.                                         4'b0001 : display_seg
    _3 = 7'b1111001;
203.                                         4'b0010 : display_seg
    _3 = 7'b0100100;
204.                                         4'b0011 : display_seg
    _3 = 7'b0110000;

```

```

205.                                     4'b0100 : display_seg
    _3 = 7'b0011001;
206.                                     4'b0101 : display_seg
    _3 = 7'b0010010;
207.                                     4'b0110 : display_seg
    _3 = 7'b0000010;
208.                                     4'b0111 : display_seg
    _3 = 7'b1111000;
209.                                     4'b1000 : display_seg
    _3 = 7'b0000000;
210.                                     4'b1001 : display_seg
    _3 = 7'b0010000;
211.                                     default : display_seg
    _3 = 7'b1111111;
212.                                     endcase
213.                                     end
214.                                2'b11:
215.                                begin
216.                                     case(display_4)
217.                                         4'b0000 : display
    _seg_4 = 7'b1000000;
218.                                         4'b0001 : display
    _seg_4 = 7'b1111001;
219.                                         4'b0010 : display
    _seg_4 = 7'b0100100;
220.                                         4'b0011 : display
    _seg_4 = 7'b0110000;
221.                                         4'b0100 : display
    _seg_4 = 7'b0011001;
222.                                         4'b0101 : display
    _seg_4 = 7'b0010010;
223.                                         4'b0110 : display
    _seg_4 = 7'b0000010;
224.                                         4'b0111 : display
    _seg_4 = 7'b1111000;
225.                                         4'b1000 : display
    _seg_4 = 7'b0000000;
226.                                         4'b1001 : display
    _seg_4 = 7'b0010000;
227.                                         default : display
    _seg_4 = 7'b1111111;
228.                                     endcase
229.                                     end
230.
231.                                endcase
232.
233.
234.                                end
235.
236.                                //time_set
237.                                else if(switch_mode == 2'b10 )
238.                                begin
239.                                    //display_4 <= clock_min_H;
240.                                    //display_3 <= clock_min_L;
241.                                    //display_2 <= clock_sec_H;
242.                                    //display_1 <= clock_sec_L;
243.                                    loop_time_set <= loop_time_set + 2'b01;
244.                                    if(loop_time_set == 2'b11)
245.                                    begin
246.                                        loop_time_set <= 2'b00;
247.                                    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.
259.                         CNT <= CNT +1;
260.                 end
261.
262.                 else if(CNT >=500 && CNT<1000)
263.                 begin
264.                         CNT <= CNT +1;
265.                         display_1 <= 4'b1111;
266.
267.                 end
268.                 else if( CNT == 1000)
269.                 CNT <= 0;
270.
271.         end
272.
273.         else if(pos_set == 2'b01)
274.         begin
275.                 display_4 <= clock_min_H;
276.                 display_3 <= clock_min_L;
277.
278.                 display_1 <= clock_sec_L;
279.
280.                 if(CNT <500)
281.                 begin
282.                         display_2 <= clock_sec_H;
283.
284.                         CNT <= CNT +1;
285.                 end
286.
287.                 else if(CNT >=500 && CNT<1000)
288.                 begin
289.                         CNT <= CNT +1;
290.                         display_2 <= 4'b1111;
291.
292.                 end
293.                 else if( CNT == 1000)
294.                 CNT <= 0;
295.
296.         end
297.
298.         else if(pos_set == 2'b10)
299.         begin
300.                 display_4 <= clock_min_H;
301.
302.                 display_2 <= clock_sec_H;
303.                 display_1 <= clock_sec_L;
304.
305.                 if(CNT <500)
306.                 begin
307.                         display_3 <= clock_min_L;
308.
309.                         CNT <= CNT +1;

```



```

302.                                     end
303.
304.                                     else if(CNT >=500 && CNT<1000)
305.                                     begin
306.                                         CNT <= CNT +1;
307.                                         display_3 <= 4'b1111;
308.
309.                                     end
310.                                     else if( CNT == 1000)
311.                                         CNT <= 0;
312.
313.                                     end
314.
315.                                     else if(pos_set == 2'b11)
316.                                     begin
317.                                         display_3 <= clock_min_L;
318.                                         display_2 <= clock_sec_H;
319.                                         display_1 <= clock_sec_L;
320.
321.                                         if(CNT <500)
322.                                         begin
323.                                             display_4 <= clock_min_H;
324.                                             CNT <= CNT +1;
325.                                         end
326.                                         else if(CNT >=500 && CNT<1000)
327.                                         begin
328.                                             CNT <= CNT +1;
329.                                             display_4 <= 4'b1111;
330.
331.                                         end
332.                                         else if( CNT == 1000)
333.                                             CNT <= 0;
334.
335.                                     end
336.
337.                                     case(loop_time_set)
338.                                     2'b00:
339.                                     begin
340.                                         case(display_1)
341.                                         4'b0000 : display_seg_1 = 7'b
342.                                         1000000;
343.                                         4'b0001 : display_seg_1 = 7'b
344.                                         1111001;
345.                                         4'b0010 : display_seg_1 = 7'b
346.                                         0100100;
347.                                         4'b0011 : display_seg_1 = 7'b
348.                                         0110000;
349.                                         4'b0100 : display_seg_1 = 7'b
350.                                         0011001;
351.                                         4'b0101 : display_seg_1 = 7'b
352.                                         0010010;
353.                                         4'b0110 : display_seg_1 = 7'b
354.                                         0000010;
355.                                         4'b0111 : display_seg_1 = 7'b
356.                                         1111000;
357.                                         4'b1000 : display_seg_1 = 7'b
358.                                         0000000;
359.                                         4'b1001 : display_seg_1 = 7'b
360.                                         0010000;

```

```

348.                                     default : display_seg_1 = 7
      'b1111111;
349.                                     endcase
350.                                     end
351.      2'b01:
352.      begin
353.                                     case(display_2)
354.                                     4'b0000 : display_seg_2 = 7'b
      1000000;
355.                                     4'b0001 : display_seg_2 = 7'b
      1111001;
356.                                     4'b0010 : display_seg_2 = 7'b
      0100100;
357.                                     4'b0011 : display_seg_2 = 7'b
      0110000;
358.                                     4'b0100 : display_seg_2 = 7'b
      0011001;
359.                                     4'b0101 : display_seg_2 = 7'b
      0010010;
360.                                     4'b0110 : display_seg_2 = 7'b
      0000010;
361.                                     4'b0111 : display_seg_2 = 7'b
      1111000;
362.                                     4'b1000 : display_seg_2 = 7'b
      0000000;
363.                                     4'b1001 : display_seg_2 = 7'b
      0010000;
364.                                     default : display_seg_2 = 7'b
      1111111;
365.                                     endcase
366.      end
367.      2'b10:
368.      begin
369.                                     case(display_3)
370.                                     4'b0000 : display_seg_3 = 7'b
      1000000;
371.                                     4'b0001 : display_seg_3 = 7'b
      1111001;
372.                                     4'b0010 : display_seg_3 = 7'b
      0100100;
373.                                     4'b0011 : display_seg_3 = 7'b
      0110000;
374.                                     4'b0100 : display_seg_3 = 7'b
      0011001;
375.                                     4'b0101 : display_seg_3 = 7'b
      0010010;
376.                                     4'b0110 : display_seg_3 = 7'b
      0000010;
377.                                     4'b0111 : display_seg_3 = 7'b
      1111000;
378.                                     4'b1000 : display_seg_3 = 7'b
      0000000;
379.                                     4'b1001 : display_seg_3 = 7'b
      0010000;
380.                                     default : display_seg_3 = 7'b
      1111111;
381.                                     endcase
382.      end
383.      2'b11:
384.      begin
385.                                     case(display_4)

```

```

386.                                4'b0000 : display_seg_4 = 7'b
    1000000;
387.                                4'b0001 : display_seg_4 = 7'b
    1111001;
388.                                4'b0010 : display_seg_4 = 7'b
    0100100;
389.                                4'b0011 : display_seg_4 = 7'b
    0110000;
390.                                4'b0100 : display_seg_4 = 7'b
    0011001;
391.                                4'b0101 : display_seg_4 = 7'b
    0010010;
392.                                4'b0110 : display_seg_4 = 7'b
    0000010;
393.                                4'b0111 : display_seg_4 = 7'b
    1111000;
394.                                4'b1000 : display_seg_4 = 7'b
    0000000;
395.                                4'b1001 : display_seg_4 = 7'b
    0010000;
396.                                default : display_seg_4 = 7'b
    1111111;
397.                                endcase
398.                                end
399.
400.                                endcase
401.
402.                                end
403. end
404.
405. end
406.
407.
408. endmodule
409.                                //ms
410.                                /*
411.                                case(loop)
412.                                2'b00:
413.                                begin
414.                                case(display_1)
415.                                4'b0000 : display_seg_1 = 7'b1000000;
416.                                4'b0001 : display_seg_1 = 7'b1111001;
417.                                4'b0010 : display_seg_1 = 7'b0100100;
418.                                4'b0011 : display_seg_1 = 7'b0110000;
419.                                4'b0100 : display_seg_1 = 7'b0011001;
420.                                4'b0101 : display_seg_1 = 7'b0010010;
421.                                4'b0110 : display_seg_1 = 7'b0000010;
422.                                4'b0111 : display_seg_1 = 7'b1111000;
423.                                4'b1000 : display_seg_1 = 7'b0000000;
424.                                4'b1001 : display_seg_1 = 7'b0010000;
425.                                default : display_seg_1 = 7'b1111111;
426.                                endcase
427.                                end
428.                                2'b01:
429.                                begin
430.                                case(display_2)
431.                                4'b0000 : display_seg_2 = 7'b1000000;
432.                                4'b0001 : display_seg_2 = 7'b1111001;
433.                                4'b0010 : display_seg_2 = 7'b0100100;
434.                                4'b0011 : display_seg_2 = 7'b0110000;

```

```

435.                                     4'b0100 : display_seg_2 = 7'b0011001;
436.                                     4'b0101 : display_seg_2 = 7'b0010010;
437.                                     4'b0110 : display_seg_2 = 7'b0000010;
438.                                     4'b0111 : display_seg_2 = 7'b1111000;
439.                                     4'b1000 : display_seg_2 = 7'b0000000;
440.                                     4'b1001 : display_seg_2 = 7'b0010000;
441.                                     default : display_seg_2 = 7'b1111111;
442.                                     endcase
443.     end
444.     2'b10:
445.     begin
446.         case(display_3)
447.             4'b0000 : display_seg_3 = 7'b1000000;
448.             4'b0001 : display_seg_3 = 7'b1111001;
449.             4'b0010 : display_seg_3 = 7'b0100100;
450.             4'b0011 : display_seg_3 = 7'b0110000;
451.             4'b0100 : display_seg_3 = 7'b0011001;
452.             4'b0101 : display_seg_3 = 7'b0010010;
453.             4'b0110 : display_seg_3 = 7'b0000010;
454.             4'b0111 : display_seg_3 = 7'b1111000;
455.             4'b1000 : display_seg_3 = 7'b0000000;
456.             4'b1001 : display_seg_3 = 7'b0010000;
457.             default : display_seg_3 = 7'b1111111;
458.         endcase
459.     end
460.     2'b11:
461.     begin
462.         case(display_4)
463.             4'b0000 : display_seg_4 = 7'b1000000;
464.             4'b0001 : display_seg_4 = 7'b1111001;
465.             4'b0010 : display_seg_4 = 7'b0100100;
466.             4'b0011 : display_seg_4 = 7'b0110000;
467.             4'b0100 : display_seg_4 = 7'b0011001;
468.             4'b0101 : display_seg_4 = 7'b0010010;
469.             4'b0110 : display_seg_4 = 7'b0000010;
470.             4'b0111 : display_seg_4 = 7'b1111000;
471.             4'b1000 : display_seg_4 = 7'b0000000;
472.             4'b1001 : display_seg_4 = 7'b0010000;
473.             default : display_seg_4 = 7'b1111111;
474.         endcase
475.     end
476. endcase
477.
478.
479.     loop <= loop + 2'b01;
480.     if(loop == 2'b11)
481.     begin
482.         loop <= 2'b00;
483.     end
484.     //
485.     //if(loop == pos_set)
486.     //begin
487.
488.
489.     //end
490.*/

```

key\_filter.v

```
1.  /*
2.      Student Name: Yang Li
3.      Studen Number: 1715977
4.  */
5.  module key_filter(  clk,
6.                      system_reset,
7.                      key_in,
8.                      key_out
9.  );
10. );
11.
12. input                clk;
13.
14. input                system_reset,
15.                      key_in;
16.
17. output               key_out;
18.
19. wire                 clk,
20.                      system_reset,
21.                      key_in;
22.
23. reg                  key_out;
24.
25.
26.
27. //    localparam TIME_20MS = 1_000_000;
28.    localparam TIME_20MS = 1000_000; //1000_000
29.
30.    reg key_cnt;
31.    reg [20:0] cnt;
32.
33.    always @(posedge clk or negedge system_reset) begin
34.        if(system_reset == 0)
35.            key_cnt <= 0;
36.        else if(cnt == TIME_20MS - 1)
37.            key_cnt <= 0;
38.        else if(key_cnt == 0 && key_out != key_in)
39.            key_cnt <= 1;
40.    end
41.
42.    always @(posedge clk or negedge system_reset) begin
43.        if(system_reset == 0)
44.            cnt <= 0;
45.        else if(key_cnt) begin
46.            if(key_out == key_in)
47.                cnt <= 0;
48.            else
49.                cnt <= cnt + 1'b1;
50.        end
51.        else
52.            cnt <= 0;
53.    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 -
103.                1'b1;
104.            else
105.                delay_cnt <= 20'd0;
106.        end
107.    end
108.
109. always@(posedge clk or negedge system_reset)
110. begin
111.     if(~system_reset)
112.     begin

```

```
113.                key_value <= 1'b1;
114.                key_flag <= 1'b0;
115.            end
116.        else
117.        begin
118.            if(delay_cnt == 20'd1)
119.            begin
120.                key_flag <= 1'b1;
121.                key_value <= key;
122.            end
123.        else
124.        begin
125.            key_flag <= 1'b0;
126.            key_value <= key_value;
127.        end
128.    end
129. end
130.end
131.always@(posedge clk)
132.begin
133.    if(key_flag && (~key_value))
134.        key_out <= 1'b0;
135.    else
136.        key_out <= 1'b1;
137.end
138.endmodule
139.
140.*/
```