**LAB REPORT**

**Laboratory exercise 3: Clocks and Timers**

3220104688 杨佳昕

26 Dec, 2024

**1. Problem Description:**

**Part I**

Build a 3-digit BCD counter displayed on HEX2-0 which counts automatically at one-second intervals after reset button is cancelled. Use KEY0 to reset the counter to 0.

**Part II**

Use HEX7-2 to build a time-of-day clock, where SW15-0 are utilized to control the initial value of hour and minute. After reset is cancelled, the clock begins to count automatically.

**Part III**

SW7-0 are used to set the time (in seconds) between reset cancel and LEDR lighting. After the game start, the set time passed and LEDR lights. The player pressed KEY3 to make LEDR out, and HEX2-0 shows the player’s reaction time in milliseconds.

**2. Design Formulation**

**Part I**

Use a variable to count how many rising edges has triggered. When reaching 50 million, i.e., one second passed, add 1 to the variable representing the HEX value. Utilize integer division and remainder expression to separate hundreds, tens and ones digit and display them on the hex display.

**Part II**

The second-counter is the same as that in Part I. When the seconds digit reaches 60, reset it to 0 and add 1 to the minutes digit. So as the minutes and hours digit (when hours digit reaches 24, just reset itself).

**Part III**

Define three states: before\_light, before\_pressed, after\_pressed.

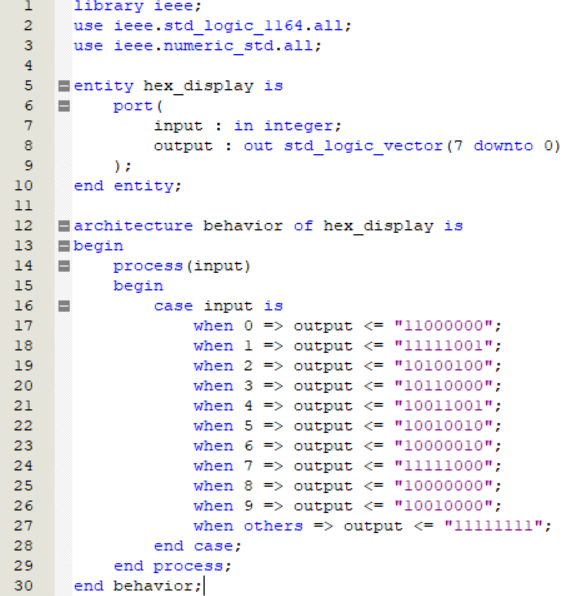
**\* Before\_light:** Before LEDR lights, copy the value of SW7-0 to a variable *light\_time*. Every time one second passes, minus 1 to *light\_time* itself. When *light\_time* runs out, light LEDR and move to *before\_pressed* state.

**\* Before\_pressed:** Before KEY3 is pressed, set counter to 0 and recount the time. Keep LEDR and counter on until KEY3 is pressed. Move to *after\_pressed* state.

**\* After\_pressed:** Make LEDR off and copy the counter value to a new variable *reaction\_time*. Separate hundreds, tens and ones digit of *reaction\_time* and display them on the hex display.

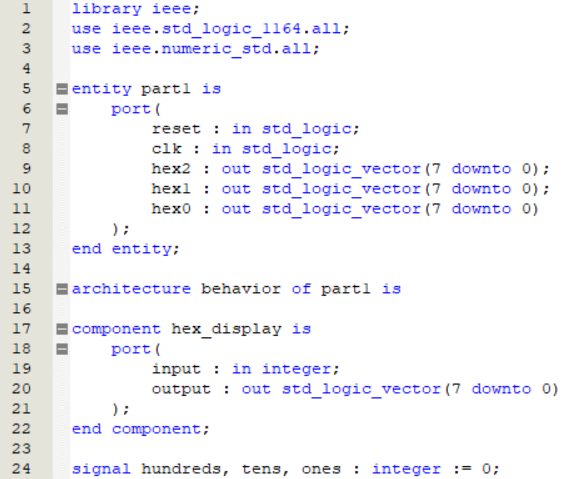
**3. Design Entry**

A VHDL file *hex\_display.vhd* to change an integer to the form in hex display:

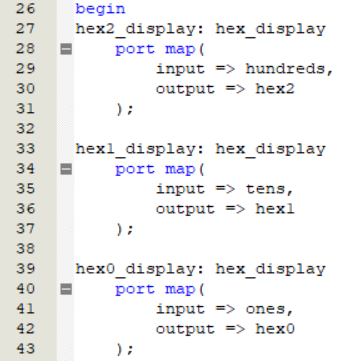


**Part I**

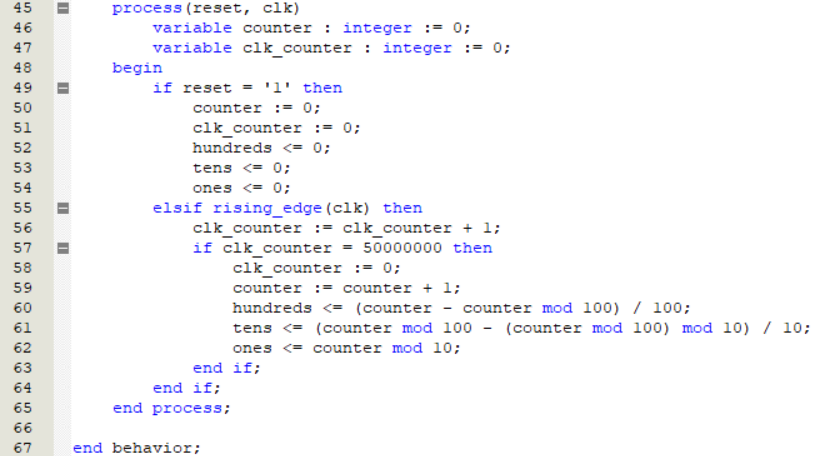
Entity, function import and signal form variable declaration:



Hex display:

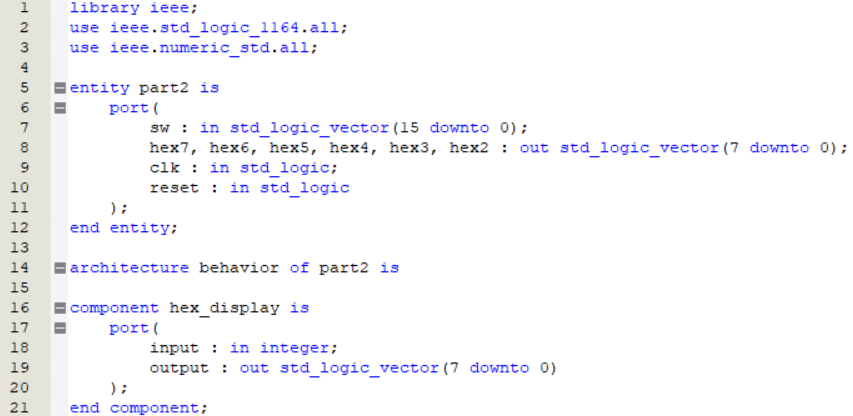


Process:

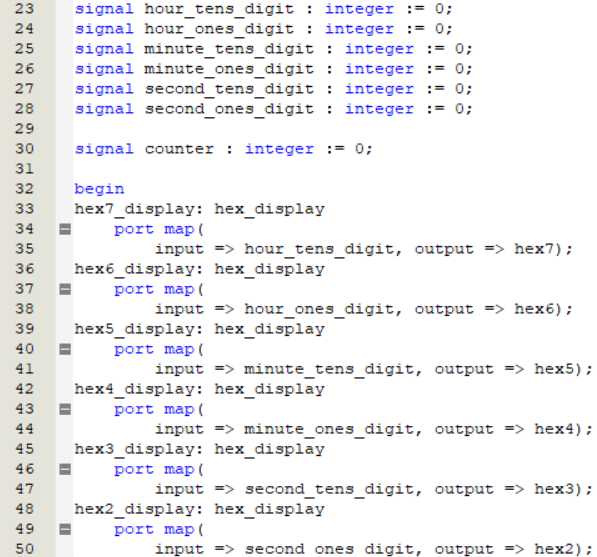


**Part II**

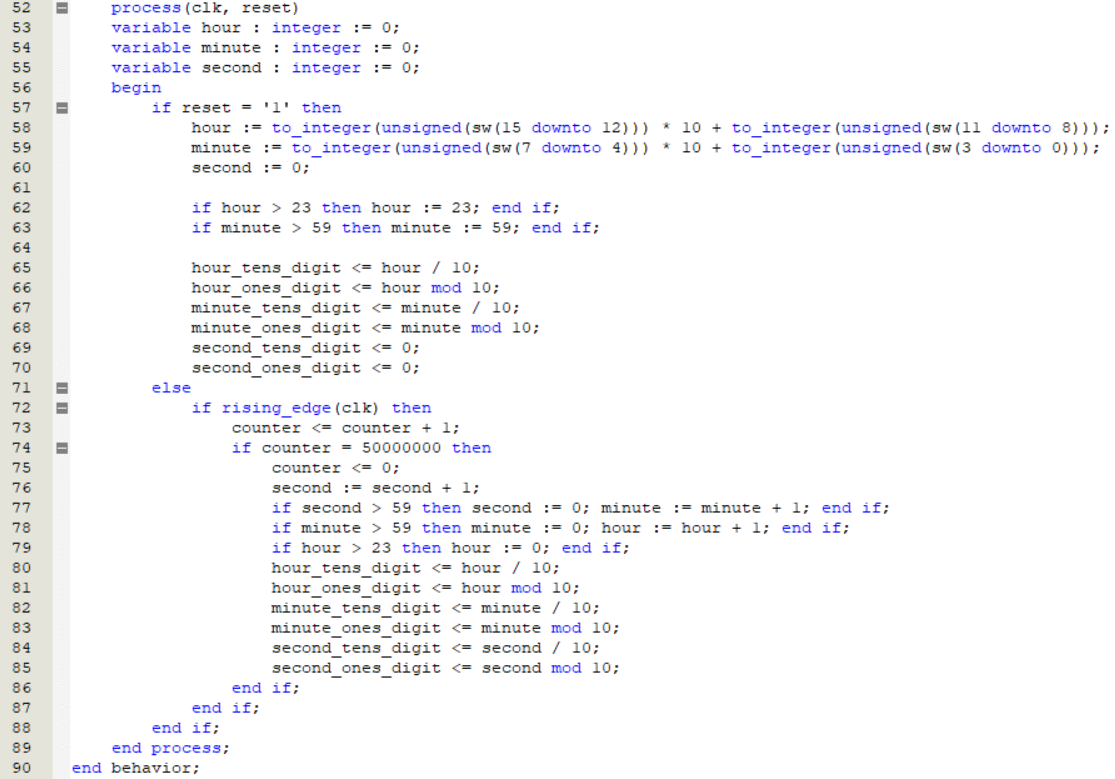
Entity and function import:



Hex display:

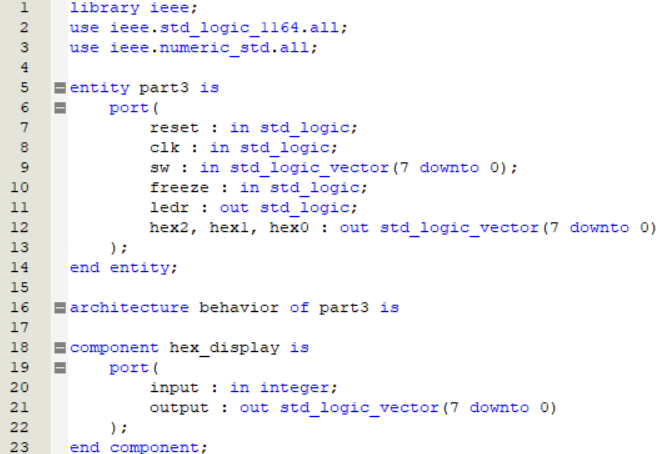


Process:

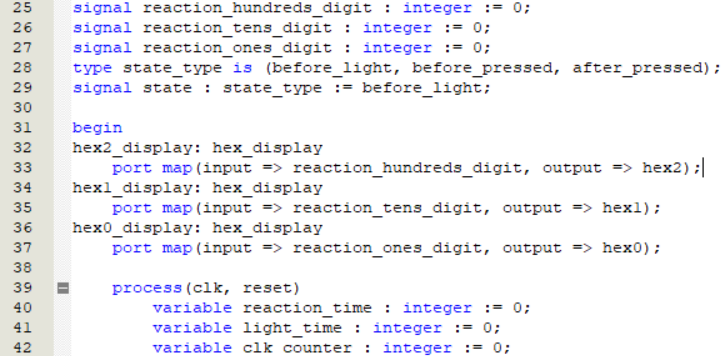


**Part III**

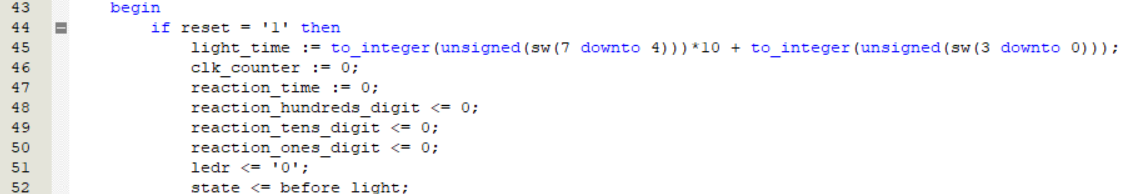
Entity and function import:



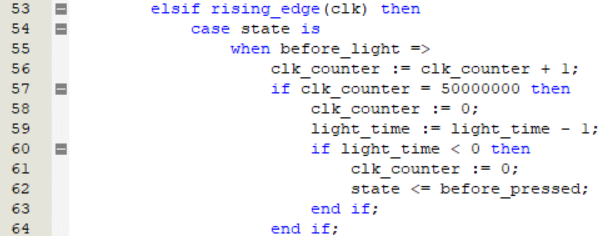
*Signal* and *variable* form variable declaration, and hex display:



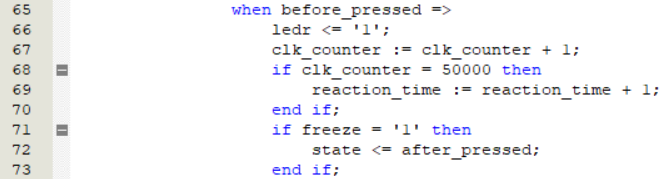
Reset:



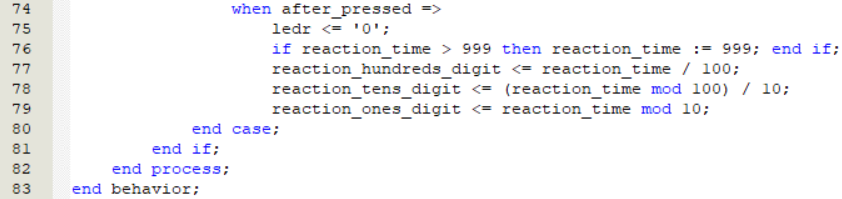
Before\_light state:



Before\_pressed state:



After\_pressed state:

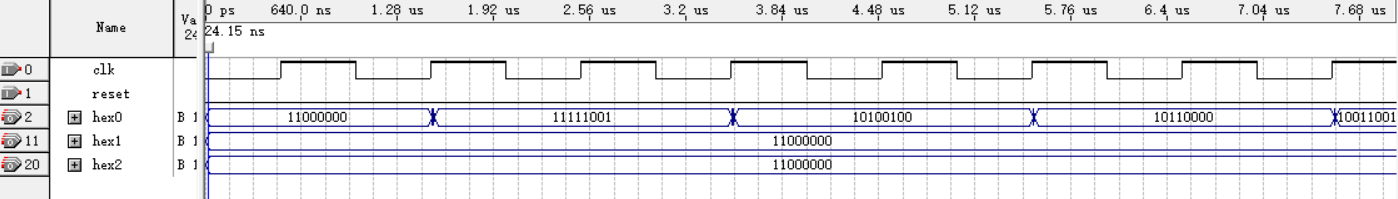


**4. Simulation Result:**

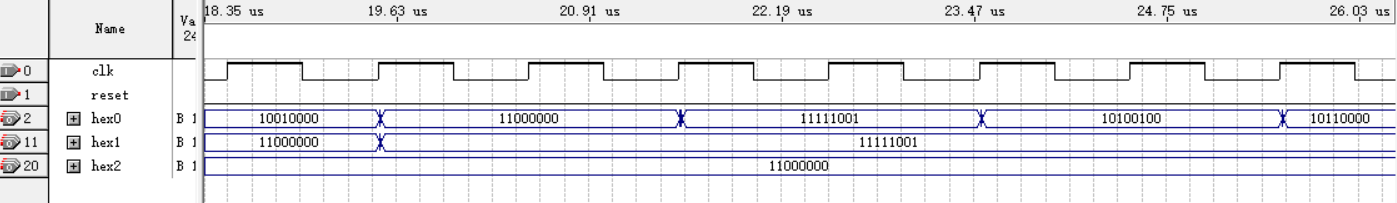
**Part I**

It is impossible to make counter work when 50 million rising edges pass in simulation, and thus I changed *50 million* to *2*.

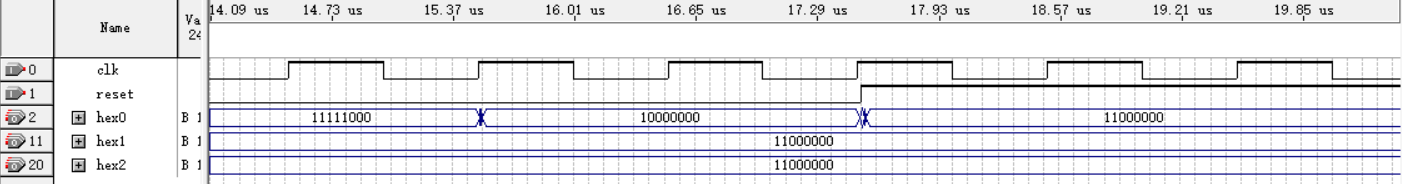
Timer:



Case:



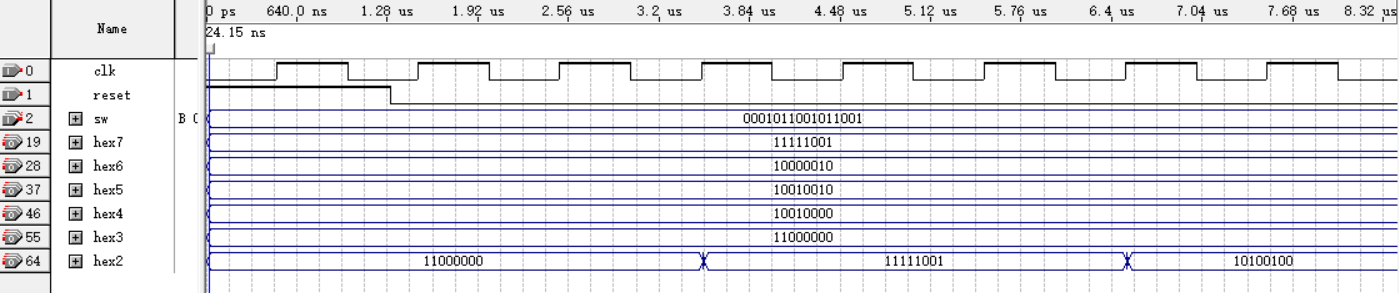
Reset:



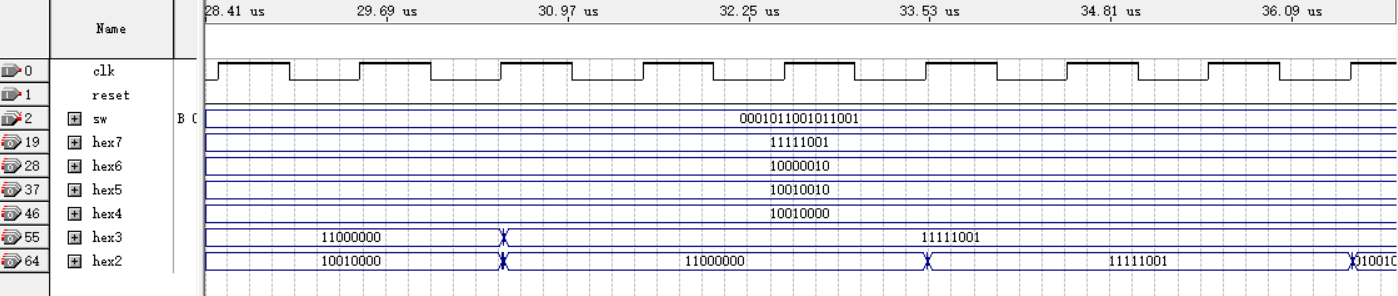
**Part II**

Change *50 million* to *2*:

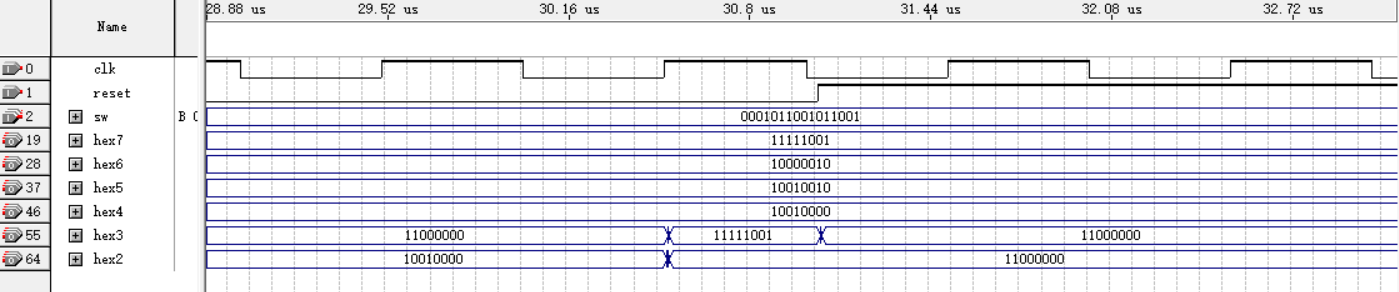
Timer:



Case:



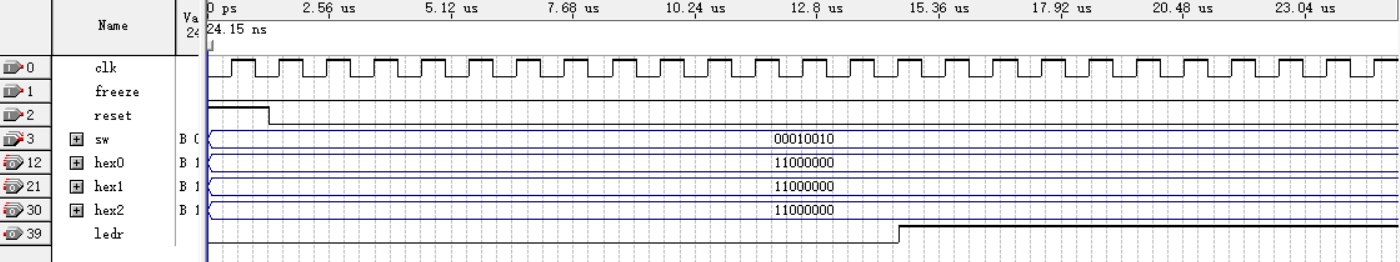
Reset:



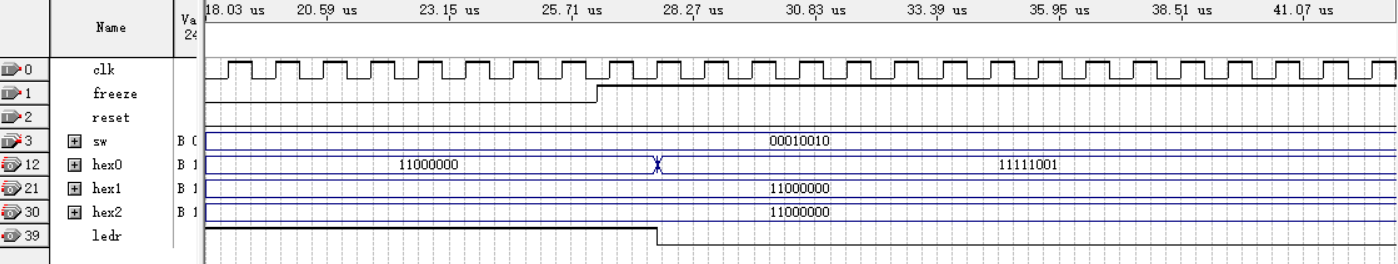
**Part III**

Change both *50 million* and *50 thousand* in the code to *1*:

LEDR lights:

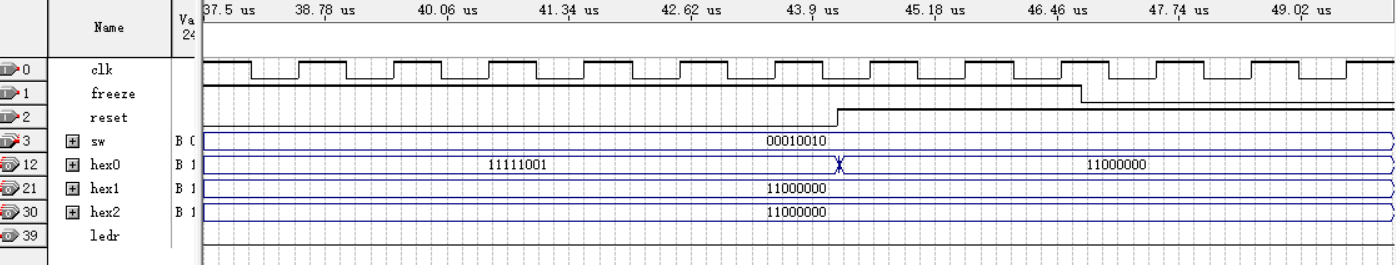


When KEY3 is pressed:



Due to the fact that the hex output is in the form of signal type variables *reaction\_cundreds\_digit*, *reactions\_tens\_digit* and *reactions\_ones\_digit*, while signal type variables do not update before the end of a process, resulting in an unavoidable delay of approximately one to two clock rising edges. However, this doesn’t matter when applicating in real, because we need 50 thousand clock rising edges to add 1 to the variable *reaction\_time* itself, which is a dramatically large number comparing with 1 or 2.

Reset:

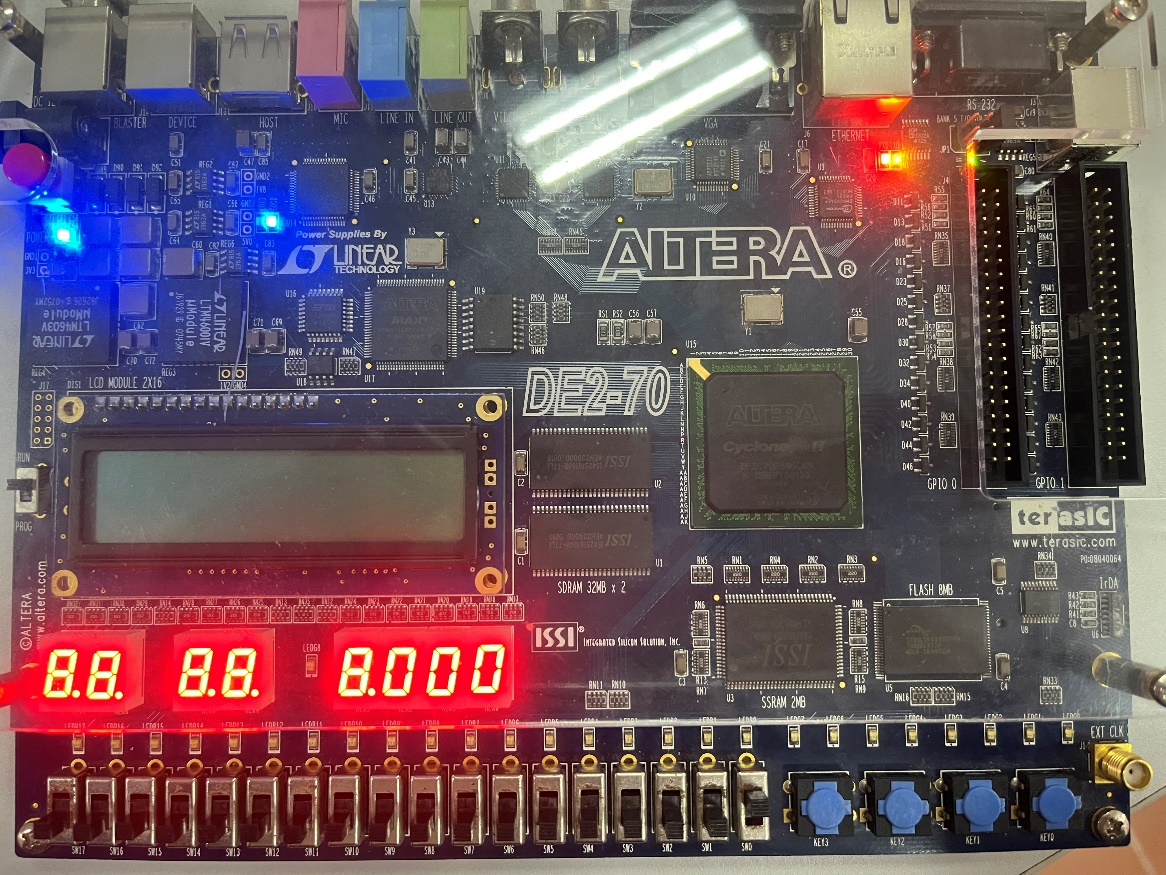


**5. Experimental Results:**

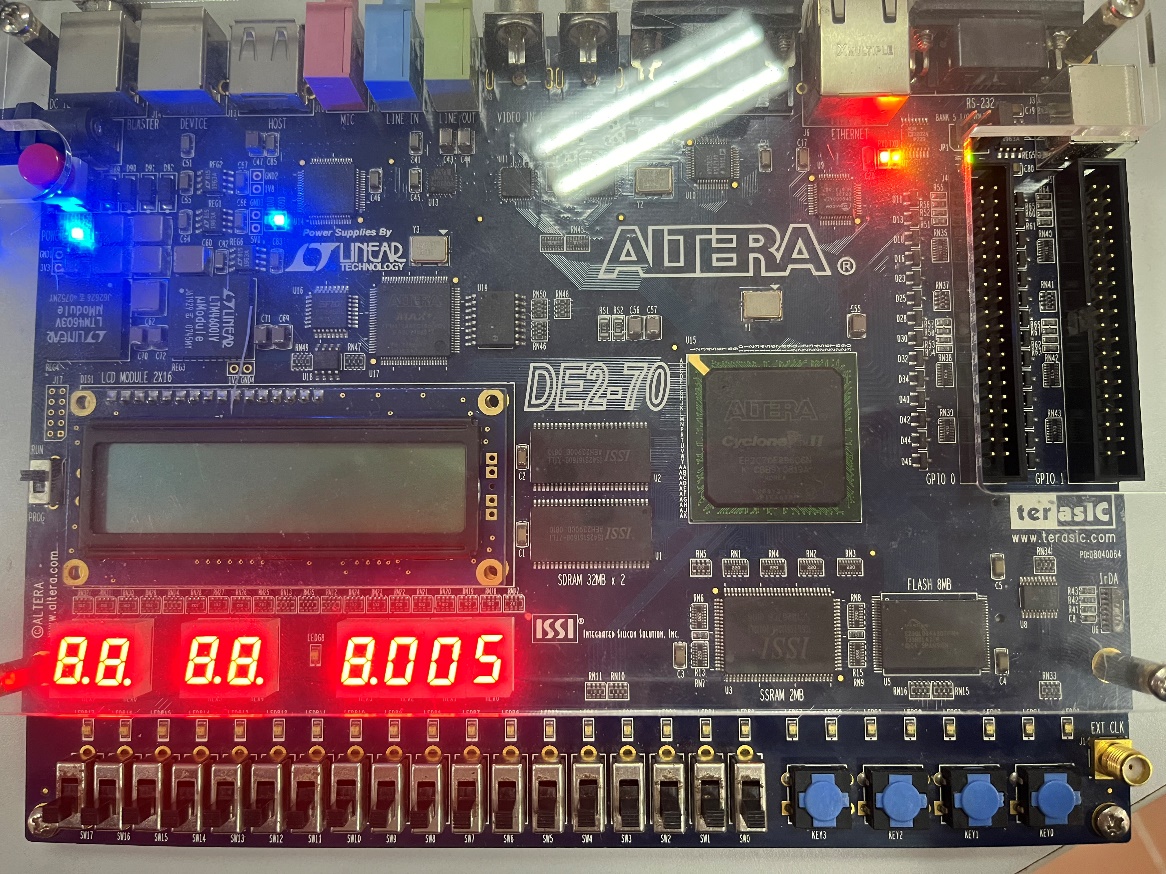
**Part I**

Note: Change KEY0 to SW0 as reset button.

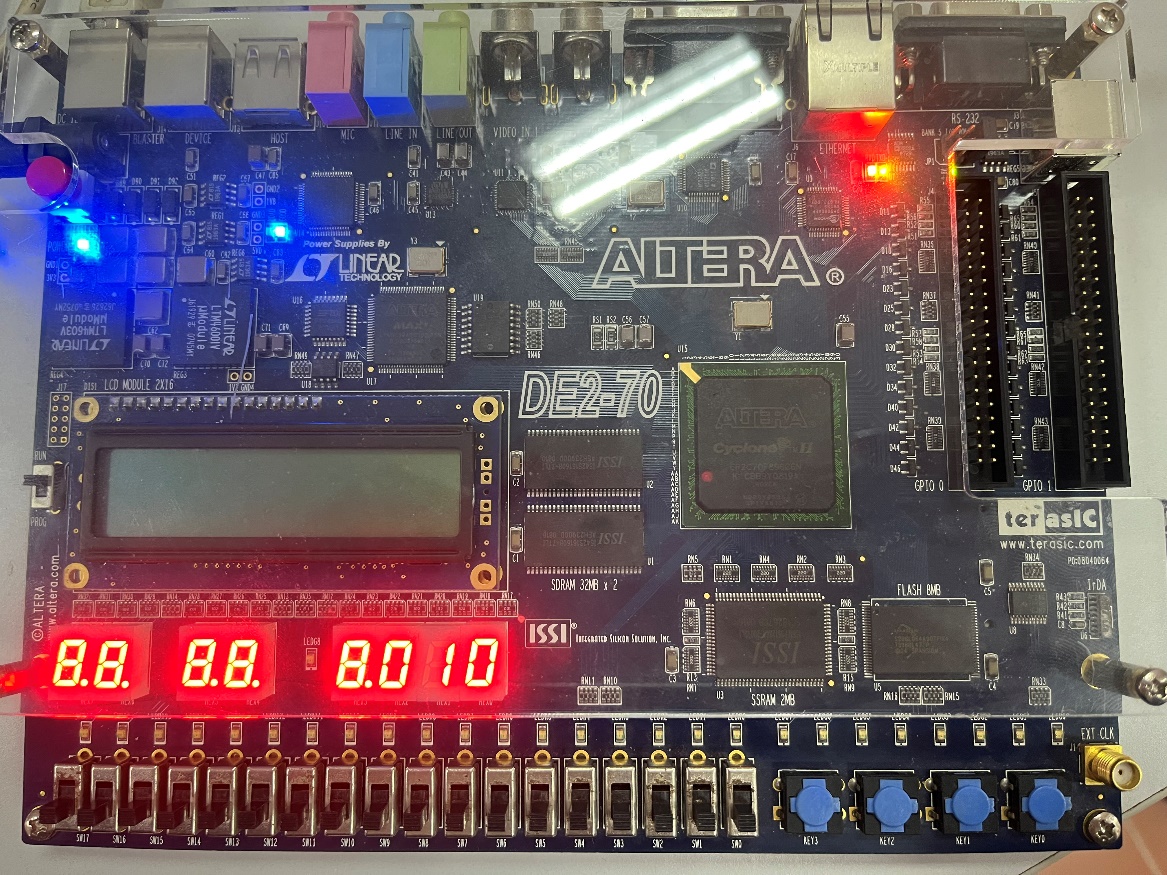
Reset:



Timer:



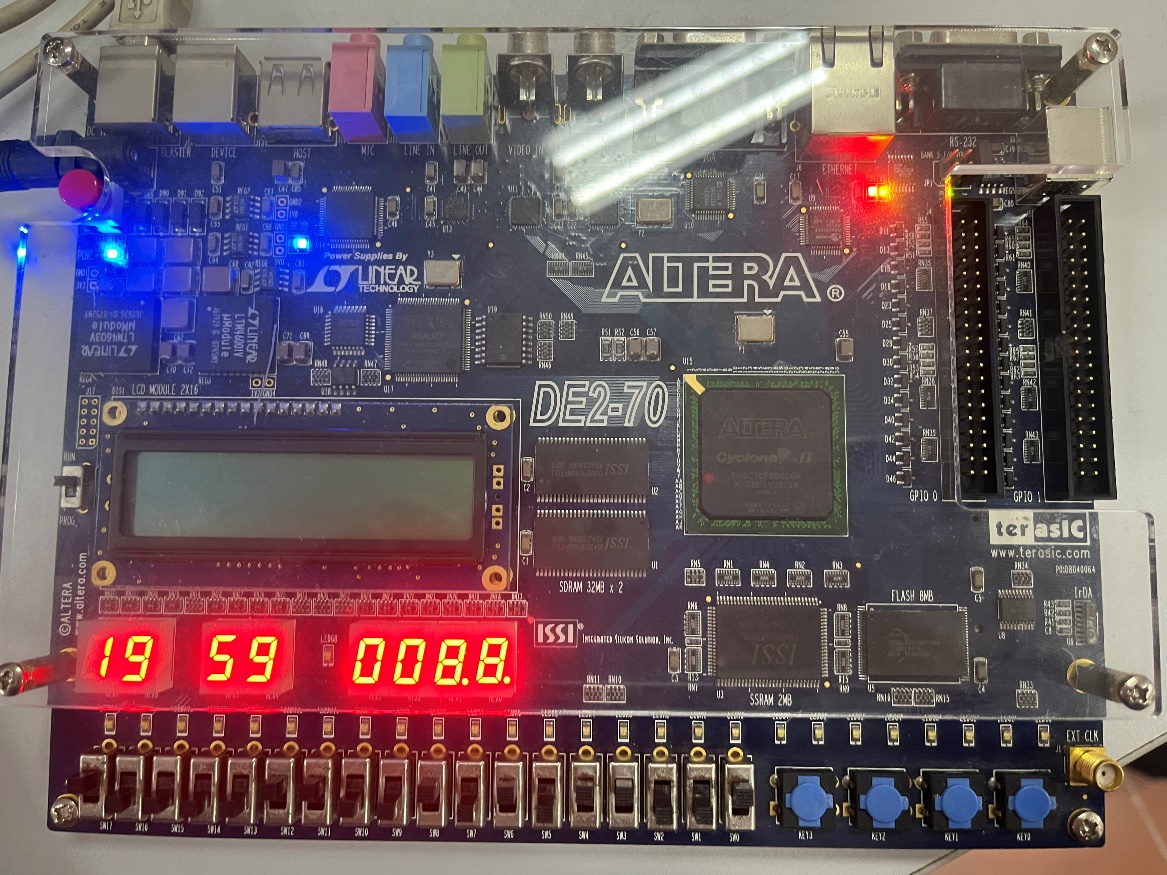
Case:



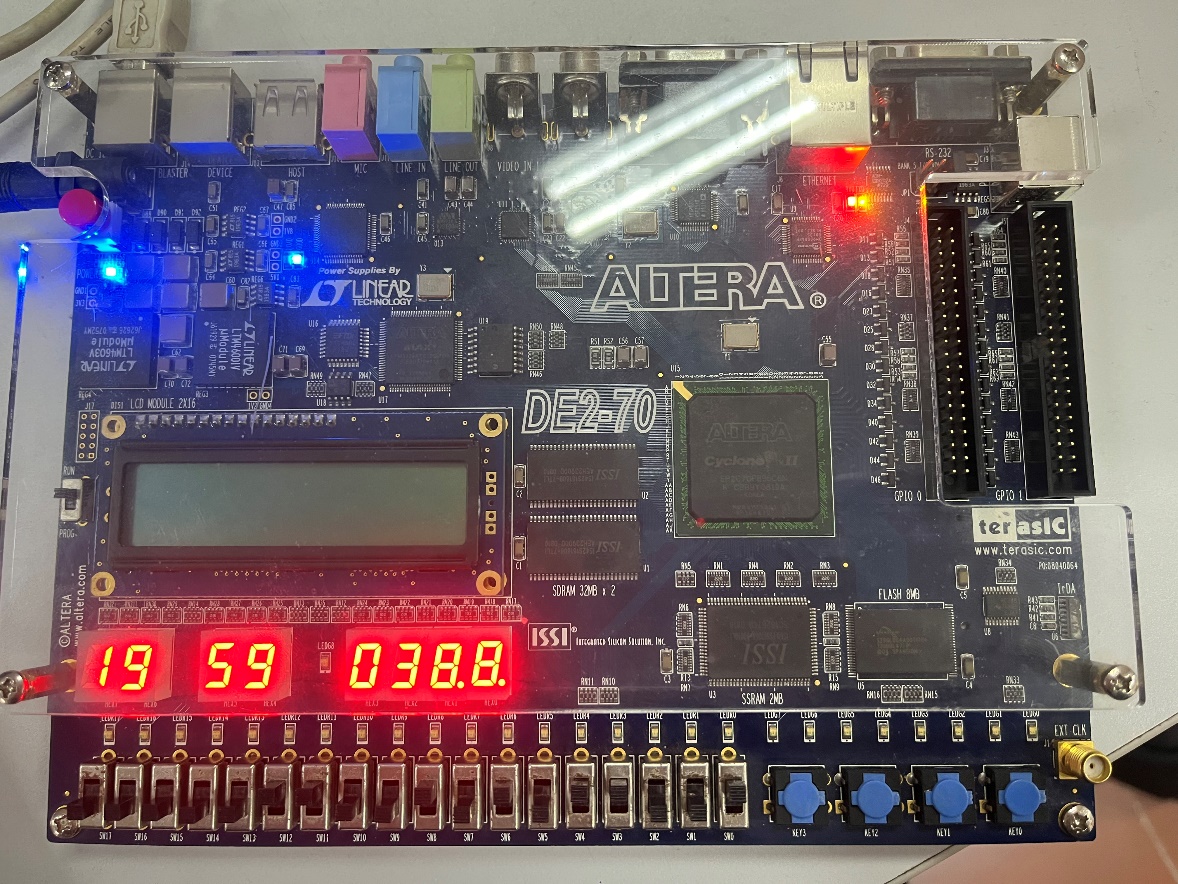
**Part II**

Start time: 19:59:00.

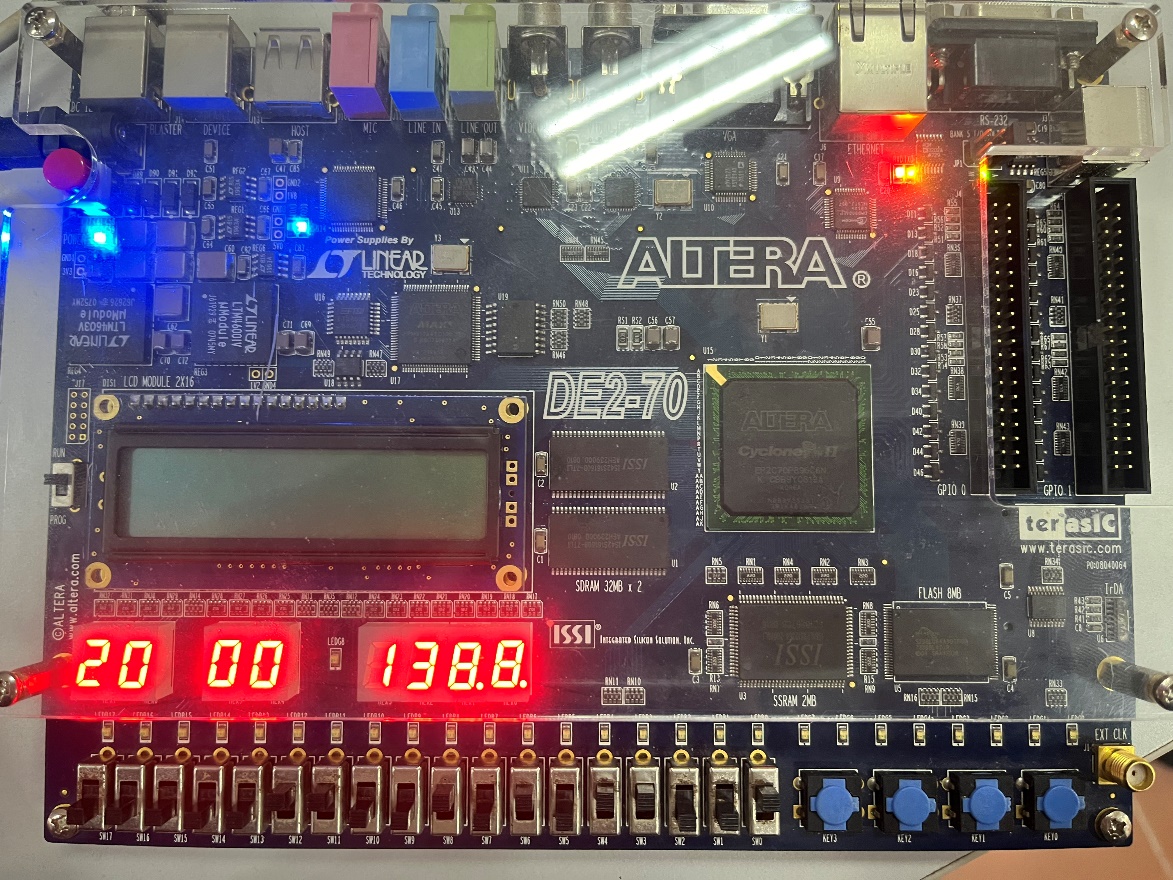
Reset (change KEY0 to SW17 as reset):



Timer:

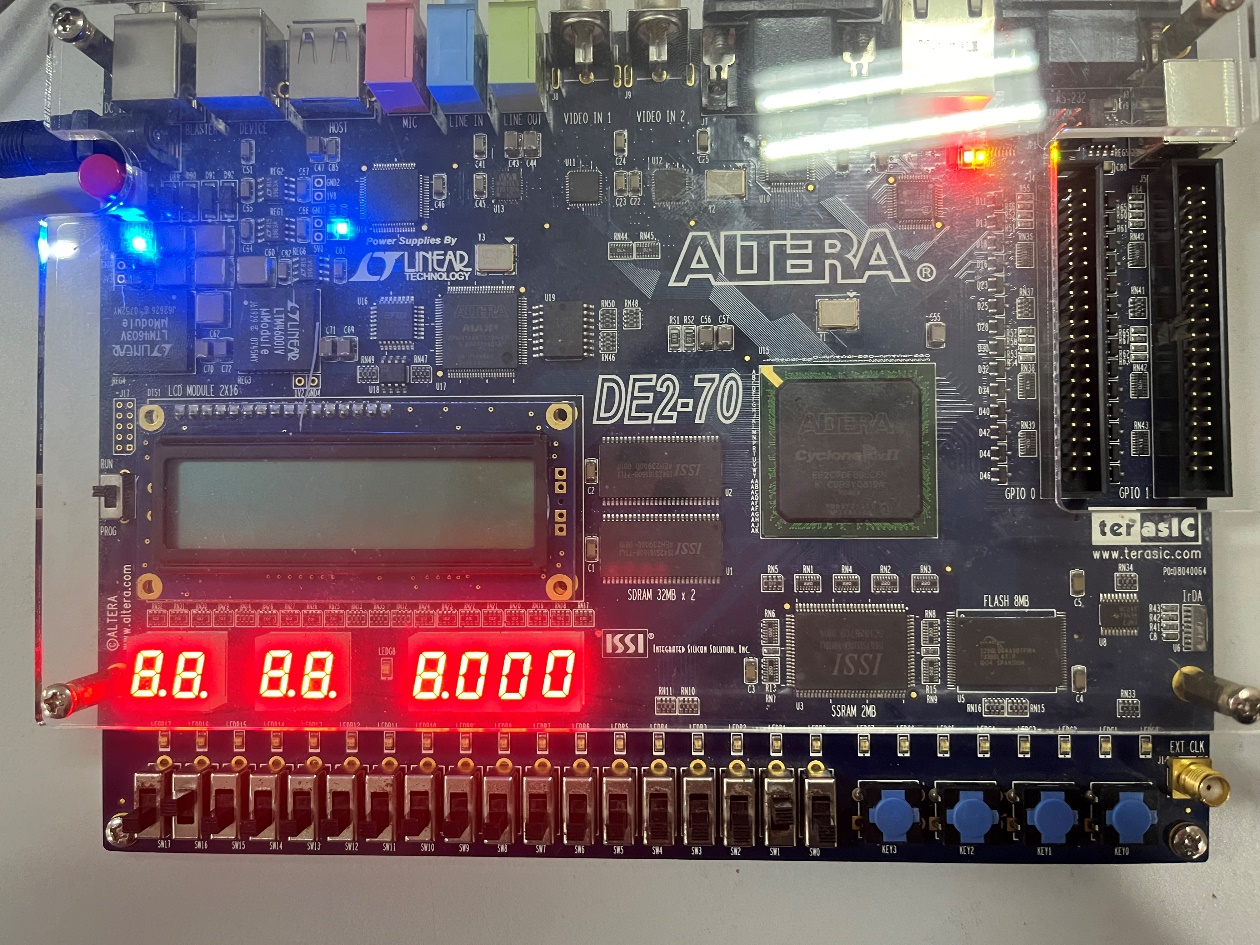


Case of second-bit, minute-bit and hour-bit:

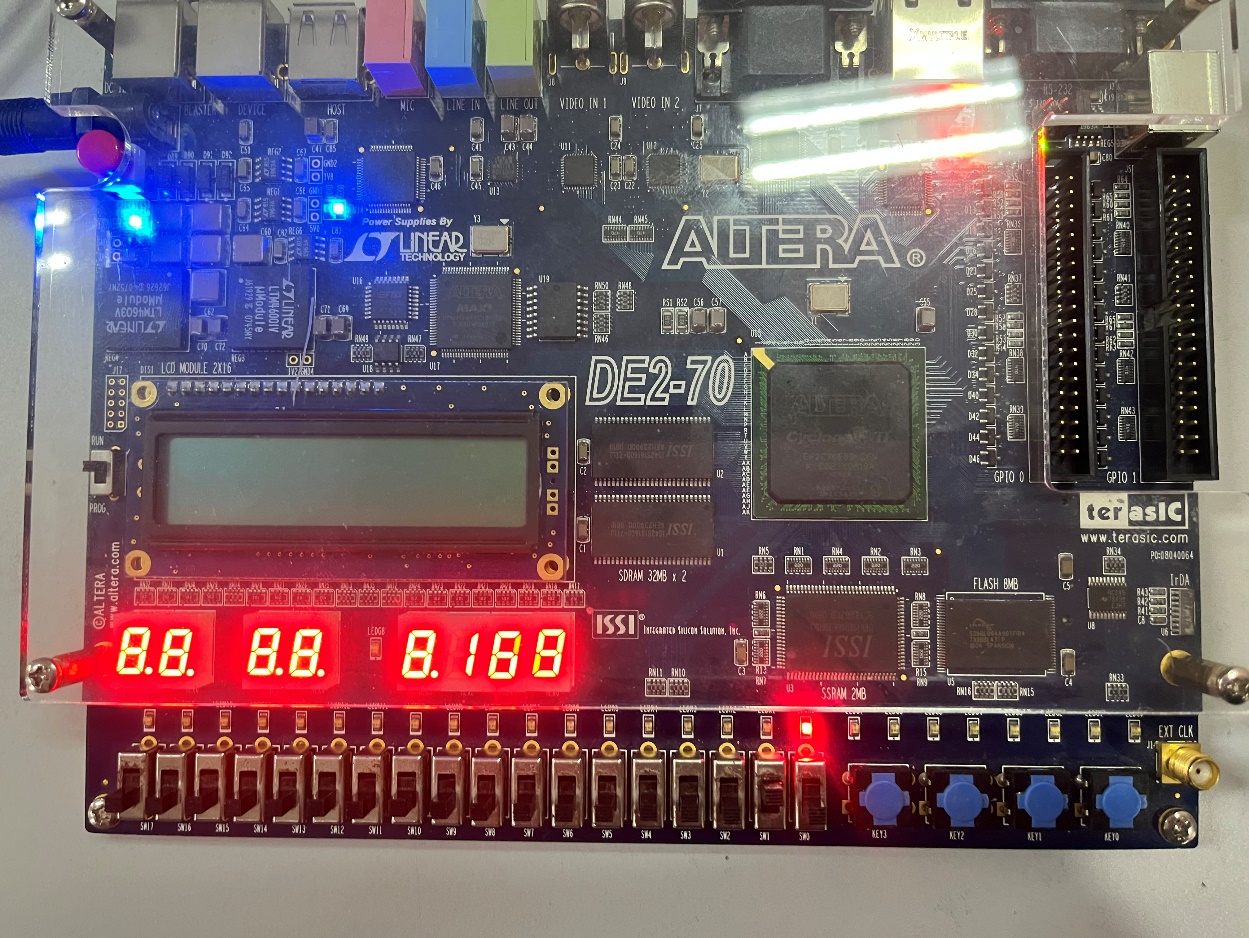


**Part III**

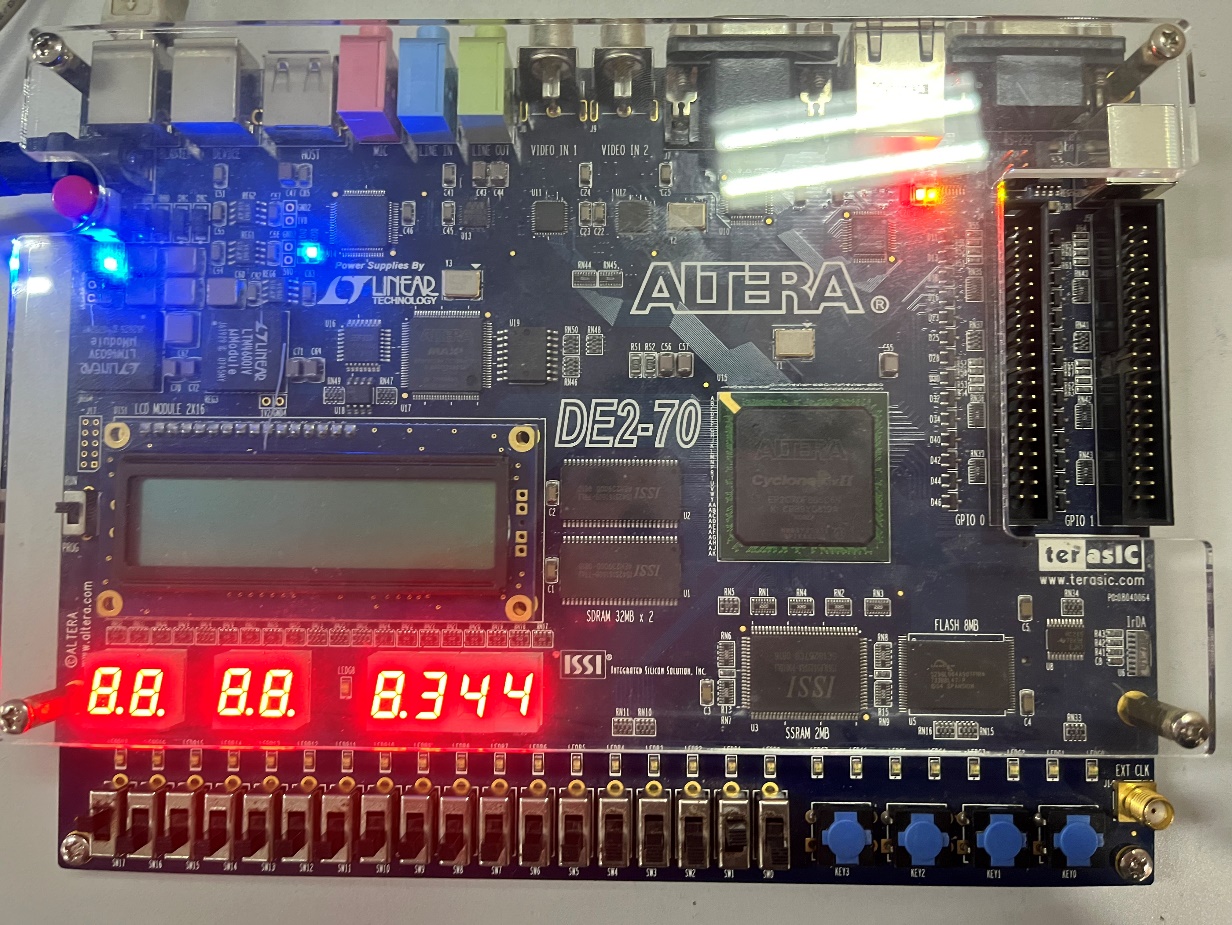
Reset (change KEY0 to SW16 as reset):



LEDR triggers:



Freeze (change KEY3 to SW17 as freezing button):

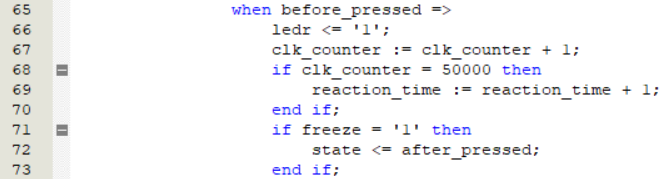


**6. Discussion and Conclusion:**

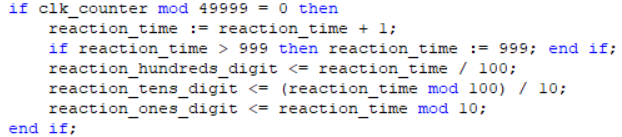
**Discussion:**

A code fault in Part III, line 68.

Change:



To:



Three mistakes have been corrected:

(1) 50000->49999: This is a tiny mistake because one clock rising edge is ignorable comparing with one second.

(2) Add code that calls back to 0 when counter reaches 50000, otherwise the if statements in the first code cannot be triggered.

(3) Add code for hex display when LEDR lights and timer runs.

**Conclusion:**

In this lab exercise on Clocks and Timers, we successfully implemented a 3-digit BCD counter, a time-of-day clock, and a reaction time game using VHDL. The BCD counter and clock displayed accurate time progression on HEX displays after respective resets. For the reaction time game, LEDR lit after a set delay, measuring player response in milliseconds. Adjustments were made to simulation parameters for practical testing. Minor code corrections improved functionality, ensuring counters reset properly and HEX displays updated accurately. This exercise demonstrated practical applications of timing circuits and reinforced VHDL programming skills.