Lab 2.2 Real-Time Clock

Zehao Chen, Xinyu Zhu

I. Goal

In this lab, we design a real time clock. The main task of the lab is to generate a real-time clock signal and a refresh signal to show the real time on the displays . The cycle of clock signal is 10 ms. With this signal, the minute, second, and 1/100 second digits change accordingly. The cycle of refresh signal is 1 ms. In one period, one of the displays is chose. The display holds for 5/6 ms and clear for 1/6 ms.

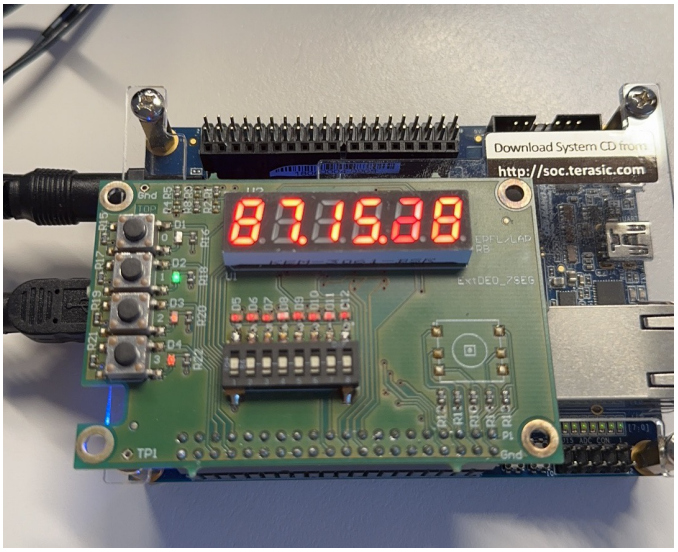

Fig. 1. The picture of real-time-clock.

II. Bloc diagram

• Whole diagram

Standing on the position of RTC, the inputs are from both Avalon bus and internal clock. The outputs will be transported to the digit displays and choose which display will be lighted and which pattern the display will show. The whole diagram is shown is figure2.

• Generate the signal with 6000 Hz

In order to generate the refresh signal, we need to design a frequency divider, which can reduce the frequency of internal clock from 50 MHz to 6000 Hz. The diminution factor is 8333. So there is a counter to record the numbers of internal clock cycle. When nReset signal is set to 0, the counter will reset to 0. When the counter counts to 8332, the counter will reset to 0, and clk_en signal will set to 1. The frequency of clk_en is 6000 Hz and the pulse width is 20 ns.

• Generate the output of Reset_Led

The cycle of the refresh signal is 1 ms, so we also need a counter here, which can reduce the frequency of clk_en from

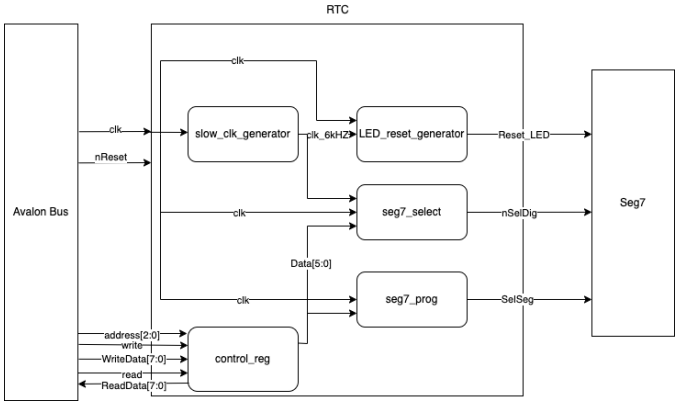

Fig. 2. The block diagram of RTC.

6000 Hz to 1000 Hz. In this process, Reset_Led signal is generated. The cycle is 1 ms and the pulse width is 1/6 ms.

• Generate the output of nSelDig

With the signal of Reset_Led_internal, which is the same with the signal of Reset_Led, we can use a counter to control the nSelDig signal. For example, when counter is 0, then nSelDig is “111110”. When counter is 1, then nSelDig is “111101”.

• Generate the output of SelSeg

Generating the SelSeg signal can be divided into 2 parts. Firstly, we need to generate the real-time clock, which means we need to design a frequency divider, which can reduce the frequency of internal clock from 50 MHz to 100 Hz. With the new clock signal, we can change the numbers which displays will show. Secondly, when nSelDig is changing, SelSeg will change to the certain pattern.

III. Register map

Address	Function
000	write WriteData[5:0] to control_reg
001-110	write WriteData[4:0] to store_reg of certain addresses
111	write WriteData[0] to rtc_flag

Fig. 3. The register map.

IV. Simulation result

As shown in the simulation wave, the clk_6kHz is generated every 8333 clk periods, and the reset_led_reg signal is refreshed every clk_6kHz periods, and the selDig_reg is increased every clk_6Khz period. And it will get back to 0

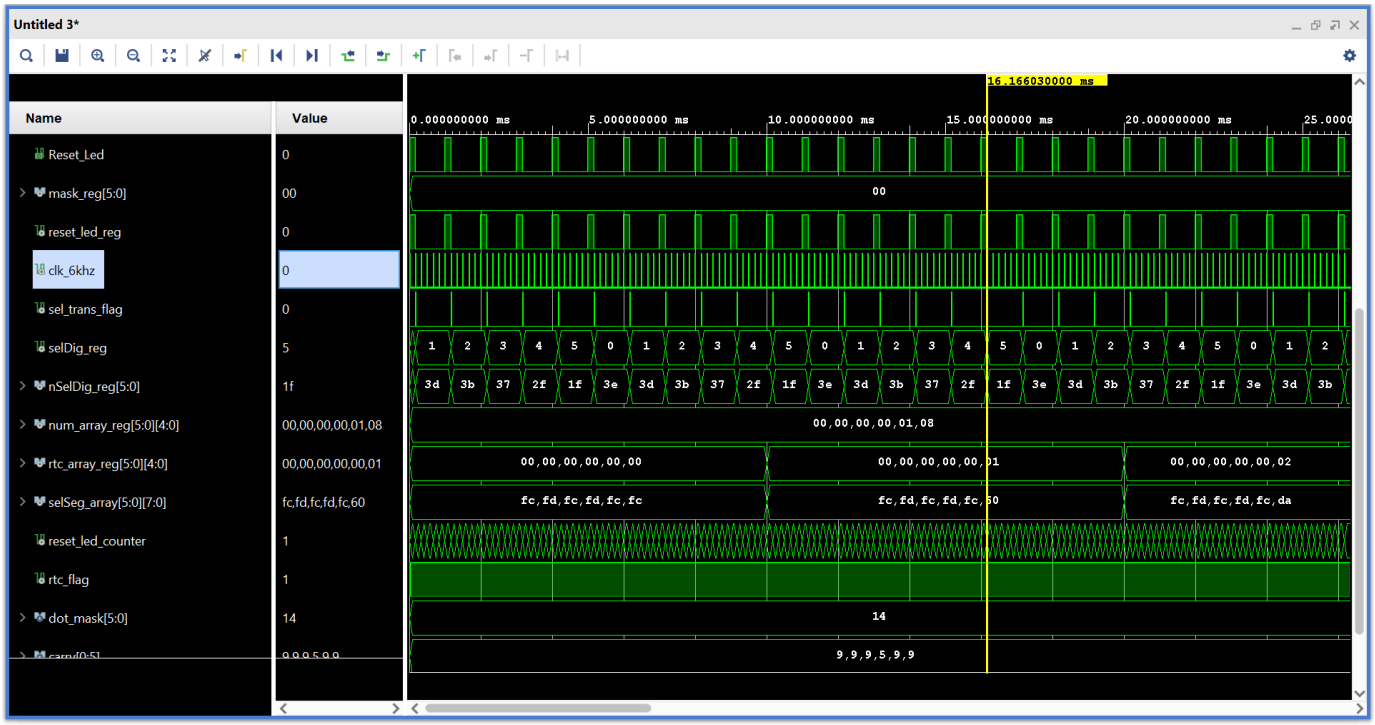

Fig. 4. The short snippet of ADC14 setting code.

when reaching 6. And for num_array_reg, its first array will increase 1 in every micro seconds, and when it reaches 9, it becomes 0 and gives a carry to next bit. And it performs well in RTC part.

For the simulation for Avalon bus, we enable write signal to 1 and write WriteData[7:0] to store_reg. And next several clk periods, we enables read to 1, the Real Time Clock component will reads the data in store_register and gives its value to Avalon bus. For the Avalon bus, the simulation runs well.

V. CONCLUSION

The code and servomotor work as expected. But there are still some precision issues.

Firstly, we accomplish the function of Real Time Clock. And our programmable display is Minutes, second, micro second. However, the function of our Real Time Clock component is just recording time, there are still some improvements of the VHDL code. For example, we are planning to add a pause function. If we press the pause button, the component will write the value of real time to a control register. And the register will continually give its value to rtc_array_reg, which is used to store value for the display number.