Lab Report 5: Seven-Segment Display
EE 102
Section 02
Bilkent University

March 27th, 2024

Doğa Zeynep Tarman
22303157

Purpose of the Experiment

The main purpose of this experiment was to design a hexadecimal counter using a seven segment display, on our FPGA boards.

Methodology

In this laboratory, I started by examining the document sent to us carefully. We were asked to design a hexadecimal counter, using the displays on our Basys 3. In order to do this, I first tried to understand which segment on the display responds to what. After understanding that, I decided which clock frequency I want to use. Having chosen the clock frequency, I defined the numbers and the letters accordingly with my VHDL code.

I then tried to answer the questions in the laboratory file, which were the following:

- What is the internal clock frequency of Basys 3?
  From what I understood, the internal clock frequency of my Basys 3 is 100MHz. I believe that this was the reason why I found it easier to implement my display with 100MHz.

- How can you create a slower clock signal from this one?
  I assume we can create such a signal using some sort of a ”divider” for the frequency. For example, we can write a VHDL code that divides the frequency by an arbitrary positive integer and resets the clock whenever we reach a value of our choice. Since after resetting the value will be 0 again, we might be able to create slower clock signals this way.

- Can you create a clock with any arbitrary frequency lower than that of the internal clock? If not, which frequencies can you create?
  I do not think this is possible. Most likely, if we can create such a thing, we can only create frequencies lower than or equal to the internal clock because of the resources and tools that we have available. But if we were to try to create a frequency higher than the internal clock, we would probably either fail to do it or need more resources than what we have on the FPGA board.

Design Specifications

In the design of my display, I decided to use two inputs and two outputs as it would be the most logical choice. As for the outputs, I made sure that I connected the right ones so that the LEDs would light up properly. In order to do this, I defined anodes and cathodes for the outputs. My code can be found in the

Appendix part of my report. In particular, mainF.vhd is my top module. c100mhx and reset are my inputs, with c100mhz being the 100mHz clock and reset being the ”clearing” input to restart the counting process.

Results

The most important part of this laboratory was to make the Basys 3 count correctly with respect to the clock input. I then ran a simulation and got a waveform as it can be seen in Figure 1.

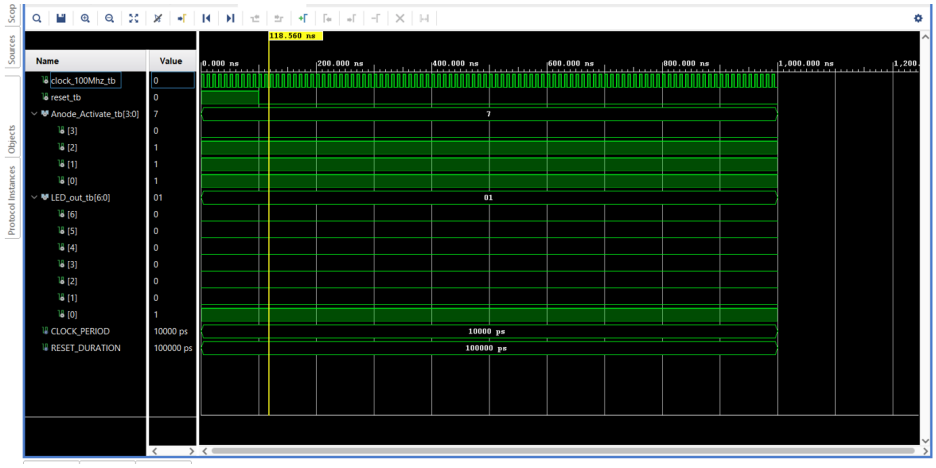

Figure 1: The Waveform

To test whether my circuit was working properly or not, I decided to look at my Basys 3 and see how the counting goes. I took a video of it, but since it is not quite possible to include the whole video here, I will try and explain my results with random pictures from the video.

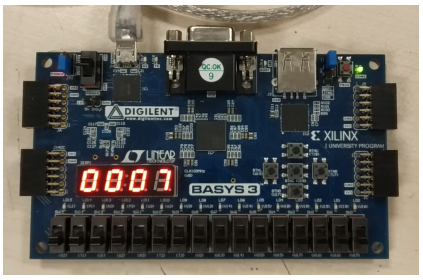

Figure 2: 7 in Hexadecimal

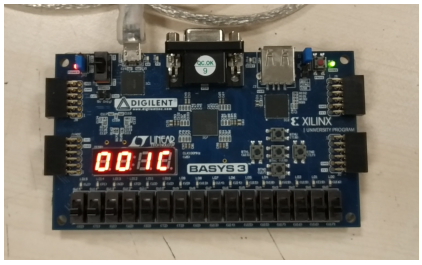

Figure 3: 28 in Hexadecimal

As it can be seen in Figure 2 and Figure 3, my Basys 3 is counting in Hexadecimal as expected.

Here are some more images from the counting process:

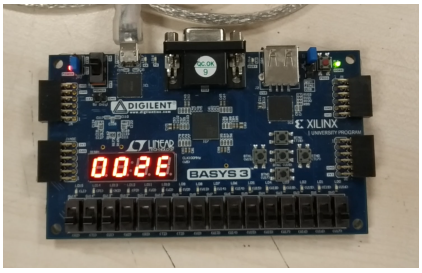

Figure 4: 46 in Hexadecimal

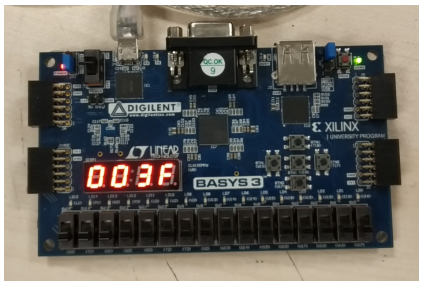

Figure 5: 63 in Hexadecimal

Again, as it can be seen in Figure 3 and Figure 4, the counting process was going smoothly.

Conclusions

In this lab, I tried to design a 7 segment display which counted in hexadecimal on VHDL. I used a clock of 100MHz, since I thought it would be easier. However, I believe that using a slower clock would be a good choice, as this high of a frequency is not necessary and the ”persistence of vision” effect is already tricking the human eye to a level in which we see the digits normally. However, I found it unnecessary since my implementation was working properly and displayed the results as expected. Nevertheless, I faced problems until I reached this ”expected” state, as I made errors due to being distracted and careless. For example, I spent an embarrassingly long amount of time to realize I forgot the test bench code, and tried to understand why my Basys 3 was not displaying things properly. Since these are all issues that I faced because of myself, I believe that overall, the laboratory was a good opportunity for me to put what we have learned in class into practice. Considering the fact that there was a seven-segment display question in our midterm, it was nice to see the display in practice as well as in theory.

Appendix

main (mainF.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity mainF is
    Port ( c100mhz : in STD_LOGIC;
           reset : in STD_LOGIC;

           aActive : out STD_LOGIC_VECTOR (3 downto 0);
           LED_out : out STD_LOGIC_VECTOR (6 downto 0));

end mainF;
architecture Behavioral of mainF is
signal oneSecCount: STD_LOGIC_VECTOR (27 downto 0);
signal oneSecEnable: std_logic;
signal daNum: STD_LOGIC_VECTOR (15 downto 0);
signal LED_BCD: STD_LOGIC_VECTOR (3 downto 0);
signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0);
signal LED_AC: std_logic_vector(1 downto 0);
begin
process(LED_BCD)
begin
    case LED_BCD is
    when "0000" => LED_out <= "0000001";
    when "0001" => LED_out <= "1001111";
    when "0010" => LED_out <= "0010010";
    when "0011" => LED_out <= "0000110";
    when "0100" => LED_out <= "1001100";
    when "0101" => LED_out <= "0100100";
    when "0110" => LED_out <= "0100000";
    when "0111" => LED_out <= "0001111";
    when "1000" => LED_out <= "0000000";
    when "1001" => LED_out <= "0000100";
    when "1010" => LED_out <= "0000010";
    when "1011" => LED_out <= "1100000";
```

```
    when "1100" => LED_out <= "0110001";
    when "1101" => LED_out <= "1000010";
    when "1110" => LED_out <= "0110000";
    when "1111" => LED_out <= "0111000";
    when others => LED_out <= "0000000";
    end case;
end process;
process(c100mhz,reset)
begin
    if(reset='1') then

        refresh_counter <= (others => '0');

    elsif(rising_edge(c100mhz)) then

        refresh_counter <= refresh_counter + 1;

    end if;
end process;
LED_AC <= refresh_counter(19 downto 18);
process(LED_AC)
begin
    case LED_AC is
    when "00" =>

        aActivate <= "0111";
        LED_BCD <= daNum(15 downto 12);

    when "01" =>

        aActivate <= "1011";
        LED_BCD <= daNum(11 downto 8);

    when "10" =>

        aActivate <= "1101";
        LED_BCD <= daNum(7 downto 4);
```

```
    when "11" =>

        aActivate <= "1110";
        LED_BCD <= daNum(3 downto 0);

    when others =>

        aActivate <= "0000";

    end case;
end process;

process(c100mhz, reset)
begin

        if(reset='1') then

            oneSecCount <= (others => '0');

        elsif(rising_edge(c100mhz)) then

            if(oneSecCount>=x"5F5E0FF") then

                oneSecCount <= (others => '0');

            else

                oneSecCount <= oneSecCount + "0000001";

            end if;

        end if;

end process;

oneSecEnable <= '1' when oneSecCount=x"5F5E0FF" else '0';

process(c100mhz, reset)
```

```
begin

        if(reset='1') then

            daNum <= (others => '0');

        elsif(rising_edge(c100mhz)) then

             if(oneSecEnable='1') then

                daNum <= daNum + x"0001";

             end if;

        end if;

end process;

end Behavioral;
```

Test Bench (testB.vhd)

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity testB is
end testB;

architecture tb_arch of testB is

    constant CLOCK_PERIOD : time := 10 ns;
    constant RESET_DURATION : time := 100 ns;

    signal clock_100Mhz_tb : std_logic := '0';
    signal reset_tb : std_logic := '0';

    signal Anode_Activate_tb : std_logic_vector(3 downto 0);
```

```
    signal LED_out_tb : std_logic_vector(6 downto 0);
begin
    DUT : entity work.mainF
        port map(
            c100mhz => clock_100Mhz_tb,
            reset => reset_tb,
            Anode_Activate => Anode_Activate_tb,
            LED_out => LED_out_tb
        );
    clock_process : process
    begin
        while now < 1000 ms loop
            clock_100Mhz_tb <= not clock_100Mhz_tb;
            wait for CLOCK_PERIOD / 2;
        end loop;
        wait;
    end process;
    reset_process : process
    begin
```

```
        reset_tb <= '1';

        wait for RESET_DURATION;
        reset_tb <= '0';

        wait for CLOCK_PERIOD / 2;

        wait;

    end process;

end tb_arch;
```