Efe Tarhan

22002840

EE102-02

30.11.2021

LAB 5: SEVEN SEGMENT DISPLAY

A) PURPOSE

Aims of this experiment are understanding perception of the human eye, displaying different numbers of digits that are connected to the same anodes and cathodes, and also simplifying the usage of seven segment display by using decoders and encoders. Another concepts and goal of this lab is understanding and implementing clock division concept.

B) DESIGN METHODOLOGY

A clock divider will be used in this lab. Basys3 has an internal clock frequency of 100Mhz. For generating a slower clock different algorithms can be used. For example a counter can count up to a desired value, n, and then start counting again. Each time the counter reaches “n”, value of the divided_clock signal will be inversed and as a result, a signal with 100Mhz/n frequency signal can be get. Also a more simple algorithm can be written. A long logic vector, n bits, can be started from the value “00000…00” and increment by one in each rising edge of the 100 Mhz clock. As a result a clock with

$\frac{100Mhz}{2^n}$ can be get. But it is important to state that only clocks with frequencies that has values 100Mhz/n where n is an integer. Because you can count only with integers, you can't increment the value of the divided_counter with decimal numbers.

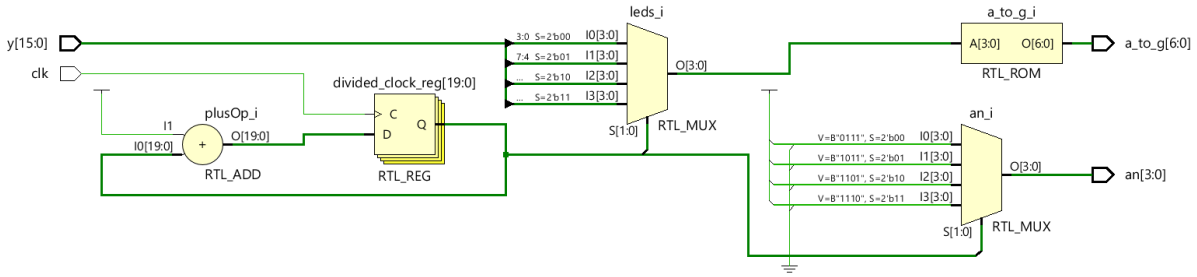
Another concept that must be emphasized is the structure of seven segment display of BASYS3. The seven segment display consists of 4 digits with each of them has 7 LEDS to be able to create hexadecimal numbers. Each digit can be turned on and of by using 4 independent anodes and 6 common cathodes. This also implies that when each digit is on ,since you choose the desired leds with the cathodes, you can't show different values on each digit at the same time. Therefore the concept of perception of human eye became important in here.

Human eye can track the light frequencies up to point which is 30-60 Hz. Therefore human eye can't track the blinking of a led with 100 Hz frequency. So, to show different values in each digit of seven segment display, each digit will bi blinked individually with a high frequency, 100Mhz/ 2^15 = 3000Hz for this project.

In this project 4 digit hexadecimal numbers will be displayed in the seven segment display by using the switches on the Basys3. 4 switch will be assigned to each digit representing each bit of the hexadecimal number. A decoder will assign codes of the cathodes that represent hexadecimal numbers to a carrier which is "leds". Than a multiplexer will be used to generate the blinking sequence of the leds and also assign the value in the input "y" to the leds.

A divided clock with 3000Hz has generated for switching between the digits.

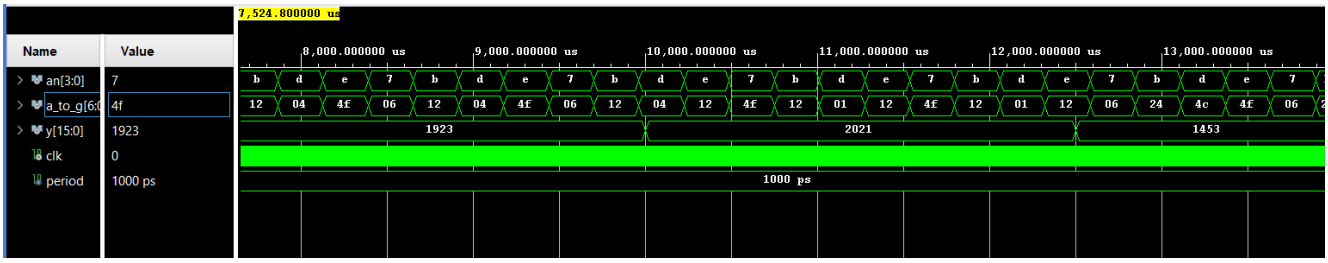
The schematic of the project can be observed below.



(Image 1: RTL design of the project)

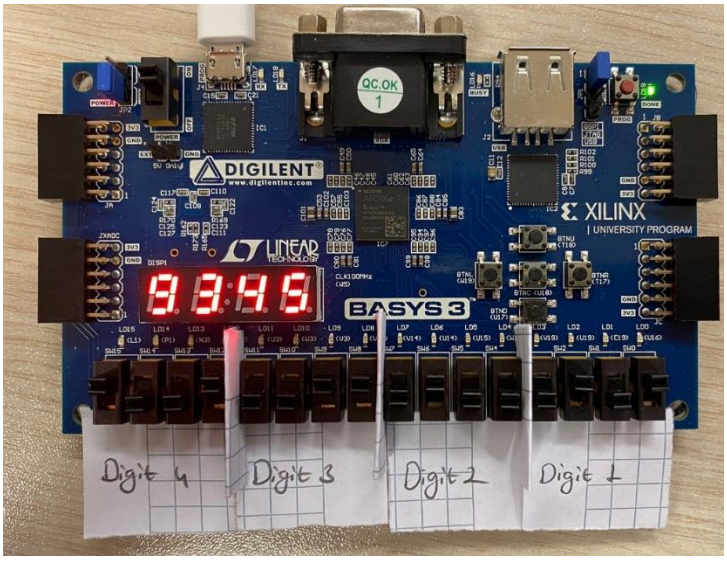
C) RESULTS

The test bench code in the appendix part has written and the simulation was done. The image below is the result of the test bench code of the seven segment displayer. Several different numbers has sent by assigning the input value , “y”, different binary values. The numbers has sent with delay of 2.5 ms and the divided_clock has a frequency of 3051 Hz therefore values has also changed fastly like expected. The result of the simulation seen below.

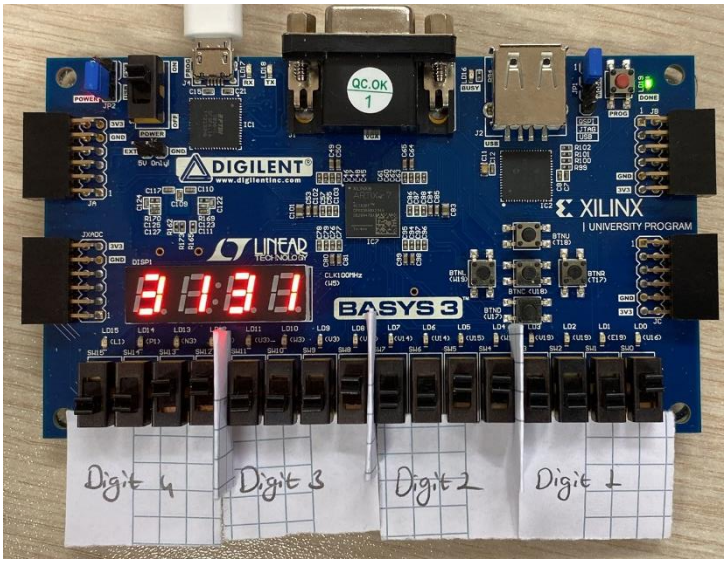


(Image 2: Waveform simulation of the test bench code of the project)

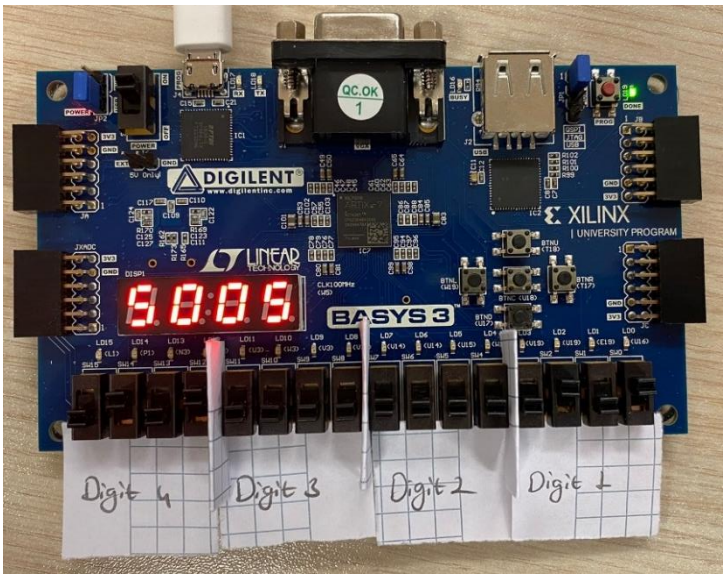
After implementing the simulation by using a test bench code it is understood that the code is working properly. Then a constraint file has written by assigning 4 switches to each digit to be able to show hexadecimal numbers. Output anodes and cathodes were assigned to the proper seven segment display constraints and bitstream has generated. After that the code has uploaded to the device and few tests has made. Tests can be seen below.



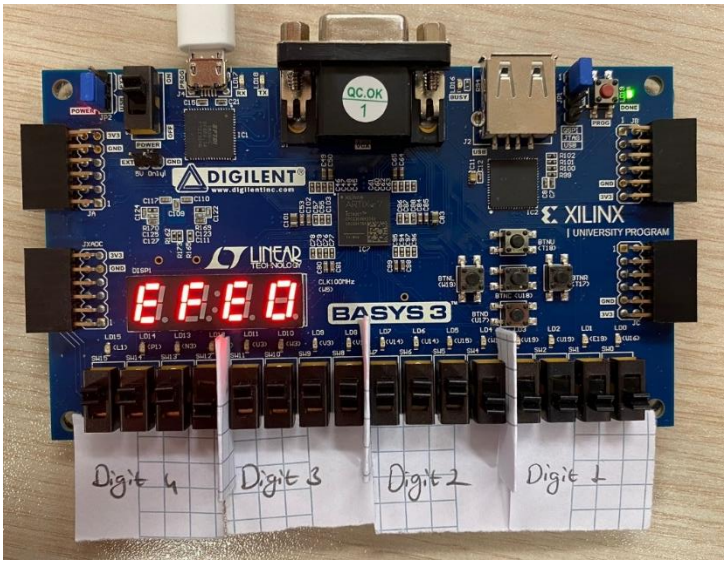
(Image 3: The seven segment displays “9345” when y = “1001001101000101”)



(Image 4: The seven segment displays “3131” when y = “0011000100110001”)



(Image 5: The seven segment displays “5005” when y = “0101000000000101”)



(Image 5: The seven segment displays “EFE0” when y = “1110111111100000”)

D) CONCLUSION

In this experiment important concepts in VHDL like clock implementation, decoder-multiplexer structures and perception frequency has practiced. Errors has occurred during writing the test bench code and the design code but they have been solved by making slight improvements with the code. This lab was useful for getting ready for preparing the term project.

E)APPENDIX

Design Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity seven_segment is
PORT(
an: out std_logic_vector(3 downto 0);
a_to_g : out std_logic_vector(6 downto 0);
y : in std_logic_vector(15 downto 0);
clk: in std_logic
);
end seven_segment;
```

```
architecture Behavioral of seven_segment is

signal divided_clock : std_logic_vector(19 downto 0) := "00000000000000000000" ;
signal activator : std_logic_vector(1 downto 0);
signal leds : std_logic_vector(3 downto 0);
begin

--Decoding 7 Segment Display Outputs
process(leds)
begin
case leds is
    when "0000" => a_to_g <= "0000001"; --0
    when "0001" => a_to_g <= "1001111"; --1
    when "0010" => a_to_g <= "0010010"; --2
    when "0011" => a_to_g <= "0000110"; --3
    when "0100" => a_to_g <= "1001100"; --4
    when "0101" => a_to_g <= "0100100"; --5
    when "0110" => a_to_g <= "0100000"; --6
    when "0111" => a_to_g <= "0001101"; --7
    when "1000" => a_to_g <= "0000000"; --8
    when "1001" => a_to_g <= "0000100"; --9
    when "1010" => a_to_g <= "0000010"; --A
    when "1011" => a_to_g <= "1100000"; --B
    when "1100" => a_to_g <= "0110001"; --C
    when "1101" => a_to_g <= "1000010"; --D
    when "1110" => a_to_g <= "0110000"; --E
    when "1111" => a_to_g <= "0111000"; --F
    when others => a_to_g <= "0000000"; --Void
end case;
end process;
```

```
--Divided Clock
process(clk)
begin
    if(rising_edge(clk)) then
        divided_clock <= divided_clock + 1;
    end if;
end process;
 activator <= divided_clock(16 downto 15);

--Multiplexing the Led choice
process(activator)
begin
case activator is
    when "00" => an <= "0111"; leds <= y(3 downto 0); -- LED 1
    when "01" => an <= "1011"; leds <= y(7 downto 4); -- LED 2
    when "10" => an <= "1101"; leds <= y(11 downto 8); -- LED 3
    when "11" => an <= "1110"; leds <= y(15 downto 12); -- LED 4
    when others => an <= "0000";
end case;
end process;

end Behavioral;
```

Test Bench Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Test_Bench is
end Test_Bench;

architecture Behavioral of Test_Bench is
```

```
COMPONENT seven_segment
Port(
an: out std_logic_vector(3 downto 0);
a_to_g : out std_logic_vector(6 downto 0);
y : in std_logic_vector(15 downto 0);
clk: in std_logic
);
end component;

signal an : std_logic_vector(3 downto 0);
signal a_to_g : std_logic_vector(6 downto 0);
signal y :  std_logic_vector(15 downto 0);
signal clk:  std_logic := '0';
constant period : time := 1ns;

begin

UUT: seven_segment port map(
an => an,
a_to_g => a_to_g,
y => y,
clk => clk
);

clock : process
begin
wait for period /2;
clk <= '1';
wait for period/2;
clk <= '0';
```

```
end process;

testBench1 : Process
begin

wait for 2.5 ms ;
y <= "0001010001010011";

wait for 2.5 ms;
y <= "0001000001110001";

wait for 2.5 ms;
y <= "0001100100100011";

wait for 2.5 ms;
y <= "0010000000100001";

end process;
end Behavioral;
```