Report: Seven Segment Display

Bilkent University Electrical and Electronics Department
EE102-01 Lab 5

Semih Akkoç - 22103132

November 22, 2022

Purpose:

The intent of this lab was to study the essentials of the seven-segment display and design a system that has the ability to achieve utilization of the seven-segment display. Additionally, getting familiar with this unit will grant a fundamental understanding of the clock module and how it operates and the implementation of this circuit on FPGA using VHDL in a modular fashion.

Questions:

What is the internal clock frequency of BASYS3?

In the BASYS3 internal default clock frequency is 100MHz and the maximum clock frequency is 450 MHz.

How can you create a slower clock signal from this one?

It can be achieved with the utilization of a clock divider so that if an N-bitted signal is defined for the clock, by using some of the initial most significant bits and writing another clock, one can achieve a slower second clock.

Can you create a clock with any arbitrary frequency lower than that of the internal clock? If not, which frequencies can you create?

As one uses a clock divider to get a slower clock, due to the fact that one uses some of the bits in the signal, one cannot use non-integer values for

clock division, which hinders the user to obtain any arbitrary frequencies. Hence, one can achieve the frequencies that are integer division -except zero- of 100MHz.

Methodology:

In order to proceed with this lab work, one needed to comprehend the presistence of vision which fundamentally relies on human eye reaction time. This idea is used in this seven-segment display lab since LEDs on display need to be turned on and off very quickly to create an illusion that they are lit simultaneously. The way the seven-segment display works can be explained as this display has seven LEDs, as the name suggests, and it has four anodes and seven cathodes to switch between LEDs. However, it is not possible to light up multiple segments synchronously. The first task was to determine which clock frequency to work with since BASYS3 has an integrated clock of 100MHz, the clock value has been chosen for this particular value. Additionally, to display to work as intended way a driver and clock modules has been implemented.

Design Specifications:

While implementing this seven-segment display, two inputs and two outputs were used, and this design's clock value was 100MHz which is sufficient enough for the persistence of vision to occur. These two inputs can be briefly explained as one being the clock, which has the described property, and the other is the reset input which resets the outputs. Fundamentally, the seven-segment displays show the seconds on display however it is shown in 16 base form. Therefore, this choice allows BASYS3 to show time for a longer period. This design's inputs and outputs can be seen in the following table.

```
CLK    : in  std_logic
RST    : in  std_logic
anode  : out std_logic_vector 4-bit long
cathode: out std_logic_vector 7-bit long
```

Subsequently, the seven-segment display module has been designed in a modular fashion. "main.vhd" is the module that utilizes the clock and driver modules. Both of those modules can be inputs, and outputs can be found in the following list respectively:

- clock.vhd

```
CLK: in std_logic;
RST: in std_logic;
counter: out std_logic_vector (27 downto 0);
div: out std_logic_vector (1 downto 0);
sec: out std_logic_vector (15 downto 0);
is_sec: out std_logic
```

- driver.vhd

```
CLK : in std_logic;
RST : in std_logic;
div : in std_logic_vector(1 downto 0); -- counter
number: in std_logic_vector(15 downto 0);
is_sec: in std_logic; --sec_en if it is 1 sec then 1
    else it is 0
anode : out std_logic_vector(3 downto 0);
cathode: out std_logic_vector(6 downto 0)
```

Results:

The suggested seven-segment displays design schematics and simulations of it, and photos of the implementation on BASYS3 can be found, and the codes regarding the modules can be found in the appendices part.

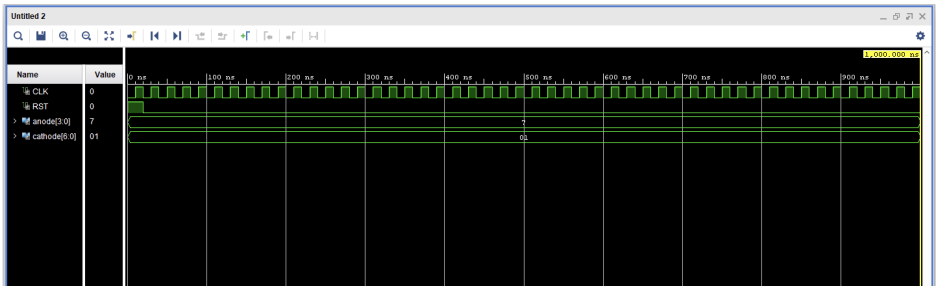


Figure 1: Simulation, main testbench of the design (main.vhd).

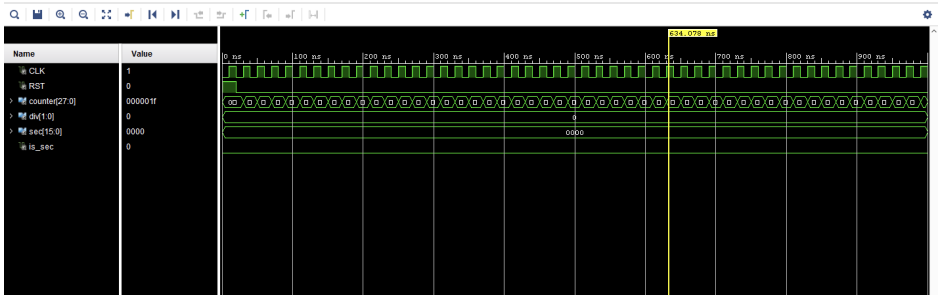


Figure 2: Simulation, clock testbench of the design (clock.vhd).

The following part includes the schematic of the seven-segment display and the top modules RLT schematic Figure 2. is on the next page.

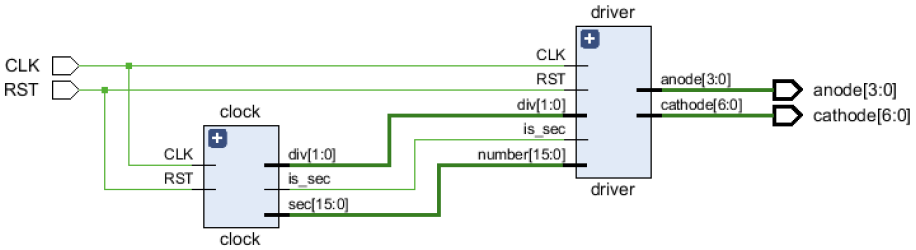


Figure 3: The small schematic.

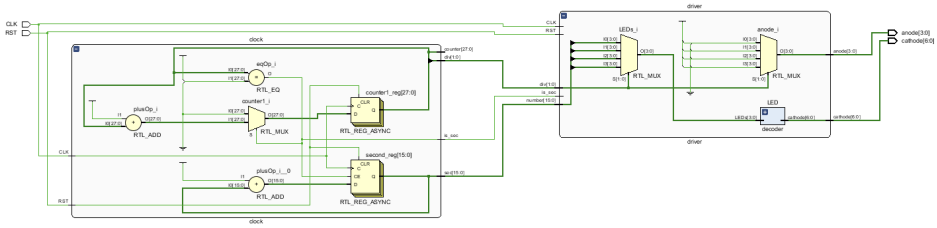


Figure 4: The big schematic.

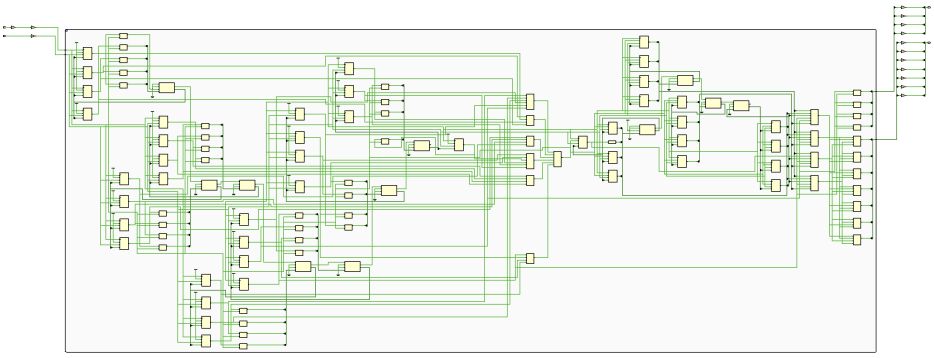
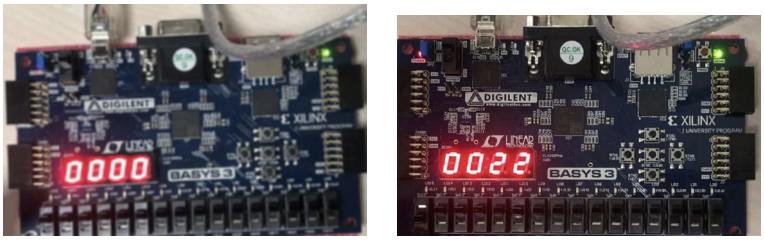


Figure 5: The RLT schematic.

In favor of assigning demonstrating the design on the BASYS3 board following images are chosen to show some of the operations done on the board. It shouldn't be forgotten that this design has used base 16.



(a) Reset condition. (b) After 34 seconds passed.

Figure 6: Working circuit on the BASYS3 board.

Conclusion:

The seven-segment display has been implemented using VHDL with the driver and the clock modules. Throughout this lab work, several concepts have been reinforced such as how a seven-segment display works and how to implement a clock and driver as well as the notion of persistence of vision.

Appendices:

Note:

Since all of the unnecessary comments in the codes have been excluded, the

remaining pages may seem few. On the contrary, by discarding the redundant comments, the real work done by the student has been brought to light which shows more quality pages than useless ones.

Code 1: (main.vhd)

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

entity main is
    Port (
        CLK: in std_logic; -- clock input
        RST: in std_logic; -- reset input
        anode: out std_logic_vector(3 downto 0);
        cathode: out std_logic_vector(6 downto 0)
       );
end main;

architecture rtl of main is

signal div: std_logic_vector(1 downto 0);
signal is_sec: std_logic;
signal number: std_logic_vector(15 downto 0);

begin

clock: entity work.clock(rtl_clock)
    PORT MAP(CLK => CLK, RST => RST, div => div, is_sec => is_sec,
        sec => number);

driver: entity work.driver(rtl_driver)
    PORT MAP(CLK => CLK, RST => RST, div => div, number => number,
        is_sec => is_sec, anode => anode, cathode => cathode);

end rtl;
```

Code 2: (clock.vhd)

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

entity clock is
    Port (
        CLK: in std_logic;
        RST: in std_logic;
        counter: out std_logic_vector(27 downto 0);
        div: out std_logic_vector(1 downto 0);
        sec: out std_logic_vector(15 downto 0);
        is_sec: out std_logic
       );
end clock;

architecture rtl_clock of clock is
    signal counter1: std_logic_vector(27 downto 0);
    signal second: std_logic_vector(15 downto 0):= (others => '0');
begin

process(CLK) begin
    if(RST = '1') then
        counter1 <= (others => '0'); second <= (others => '0');
    else
        if rising_edge(CLK) then
            counter1 <= counter1 + '1';
                if counter1 =x"5F5E0FF" then
                    second <= second + '1';
                    counter1 <= (others => '0');
                end if;
        end if;
    end if;
end process;

is_sec <= '1' when counter1 =x"5F5E0FF" else '0';
div <= counter1(19 downto 18);
counter <= counter1;
sec <= second;
```

```
end rtl_clock;
```

Code 3: (driver.vhd)

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


entity driver is
  Port (
    CLK : in std_logic;
    RST : in std_logic;
    div : in std_logic_vector(1 downto 0); -- phase counter
    number: in std_logic_vector(15 downto 0);
    is_sec: in std_logic; --sec_en if it is 1 sec then 1 else it is
        0
    anode : out std_logic_vector(3 downto 0);
    cathode: out std_logic_vector(6 downto 0)
   );
end driver;

architecture rtl_driver of driver is

signal LEDs: std_logic_vector(3 downto 0);

begin

process(div) begin
    case div is
        when "00" => anode <= "0111"; -- 1st digit
            LEDs <= number(15 downto 12);

        when "01" => anode <= "1011"; -- 2nd digit
            LEDs <= number(11 downto 8);

        when "10" => anode <= "1101"; -- 3rd digit
            LEDs <= number(7 downto 4);
```

```
        when "11" => anode <= "1110"; -- 4th digit
            LEDs <= number(3 downto 0);

        when others => anode <= "1111";
    end case;
end process;

--LED: decoder PORT MAP(LEDs=>LEDs, cathode => cathode);
LED: entity work.decoder(rtl_decoder)
    PORT MAP(LEDs=>LEDs, cathode => cathode);

end rtl_driver;
```

Code 4: (decoder.vhd)

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

entity decoder is
    Port (
        LEDs : in std_logic_vector(3 downto 0);
        cathode : out std_logic_vector(6 downto 0)
     );
end decoder;

architecture rtl_decoder of decoder is

begin

process(LEDs) begin
    case LEDs is
        when "0000" => cathode <= "0000001"; --0
        when "0001" => cathode <= "1001111";
        when "0010" => cathode <= "0010010";
        when "0011" => cathode <= "0000110";
        when "0100" => cathode <= "1001100";
        when "0101" => cathode <= "0100100";
        when "0110" => cathode <= "0100000";
        when "0111" => cathode <= "0001111";
        when "1000" => cathode <= "0000000";
        when "1001" => cathode <= "0000100";
```

```
        when "1010" => cathode <= "0000010";
        when "1011" => cathode <= "1100000";
        when "1100" => cathode <= "0110001";
        when "1101" => cathode <= "1000010";
        when "1110" => cathode <= "0110000";
        when "1111" => cathode <= "0111000"; --F
        when others => cathode <= "1111111"; --off
    end case;
end process;

end rtl_decoder;
```

Code 5: (main_testbench.vhd)

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

entity main_testbench is
--  Port ( );
end main_testbench;

architecture tb of main_testbench is

signal CLK: std_logic;
signal RST: std_logic;
signal anode: std_logic_vector(3 downto 0);
signal cathode: std_logic_vector(6 downto 0);

begin

dut: entity work.main(rtl)
    PORT MAP (CLK => CLK, RST=>RST, anode=>anode, cathode=>cathode);

clock_process : process begin
CLK <= '0';
wait for 10 ns;
CLK <= '1';
wait for 10 ns;
end process;
stim_proc: process begin
RST <= '1';
```

```
wait for 20 ns;
RST <= '0';
wait;
end process;

end tb;
```

Code 6: (clock_testbench.vhd)

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

entity clock_testbench is
--  Port ( );
end clock_testbench;

architecture tb_clock of clock_testbench is

signal CLK: std_logic;
signal RST: std_logic;
signal counter: std_logic_vector(27 downto 0);
signal div: std_logic_vector(1 downto 0);
signal sec: std_logic_vector(15 downto 0);
signal is_sec: std_logic;

begin
dut: entity work.clock
    PORT MAP( CLK => CLK, RST => RST, counter => counter, div =>
        div, sec => sec, is_sec => is_sec);

clock_process: process begin
clk <= '0';
wait for 10ns;
clk <= '1';
wait for 10ns;
end process;

stim_proc: process begin
RST <= '1';
wait for 20 ns;
RST <= '0';
```

```
wait;
end process;

end tb_clock;
```

References:

- "Persistence of vision." Wikipedia, Wikimedia Foundation, 21 Nov. 2022, en.wikipedia.org/wiki/Persistence_of_vision. Accessed 22 Nov. 2022.
- https://github.com/CankutBoraTuncer/Bilkent-EEE102-Labs