Muhammed Enes İnanç

22001799

24.11.2022

Purpose:

The aim of this lab is using the seven-segment display on BASYS3. Also we find a chance to use our knowledge about decoder, multiplexer and hierarchical modules.

Questions:

What is the internal clock frequency of BASYS3?

BASYS3 has an internal clock which frequency is 100 MHz.

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

By using a clock divider it can be done. If a n bit signal is implemented for the clock, and it can be designed an another clock by using some of the most significant bits of this signal.

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

An arbitrary reduction is not possible. bits travel in powers of 2 and decrease in power of 2 unless we use one or both of these bits.

Methodology:

In this lab, we also examined how the human eye continuously sees high-frequency lights. Also, the reason why this led panel on basys3 is called 7 segments is that it consists of 7 led bars and a dot. Moreover, it has four anodes and seven cathodes to switch between these LEDs and activate different LEDs. The first task is that BASYS3 has an integrated 100MHz clock, which is also suitable for viewing by the human eye. In this project, a driver and clock module was created.

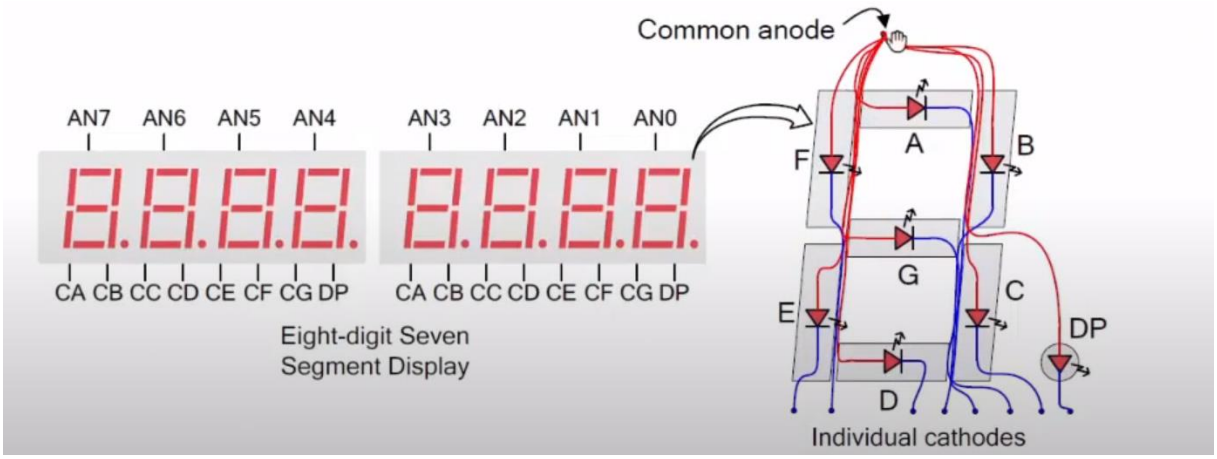

Figure 1 (How does 7 segment work)

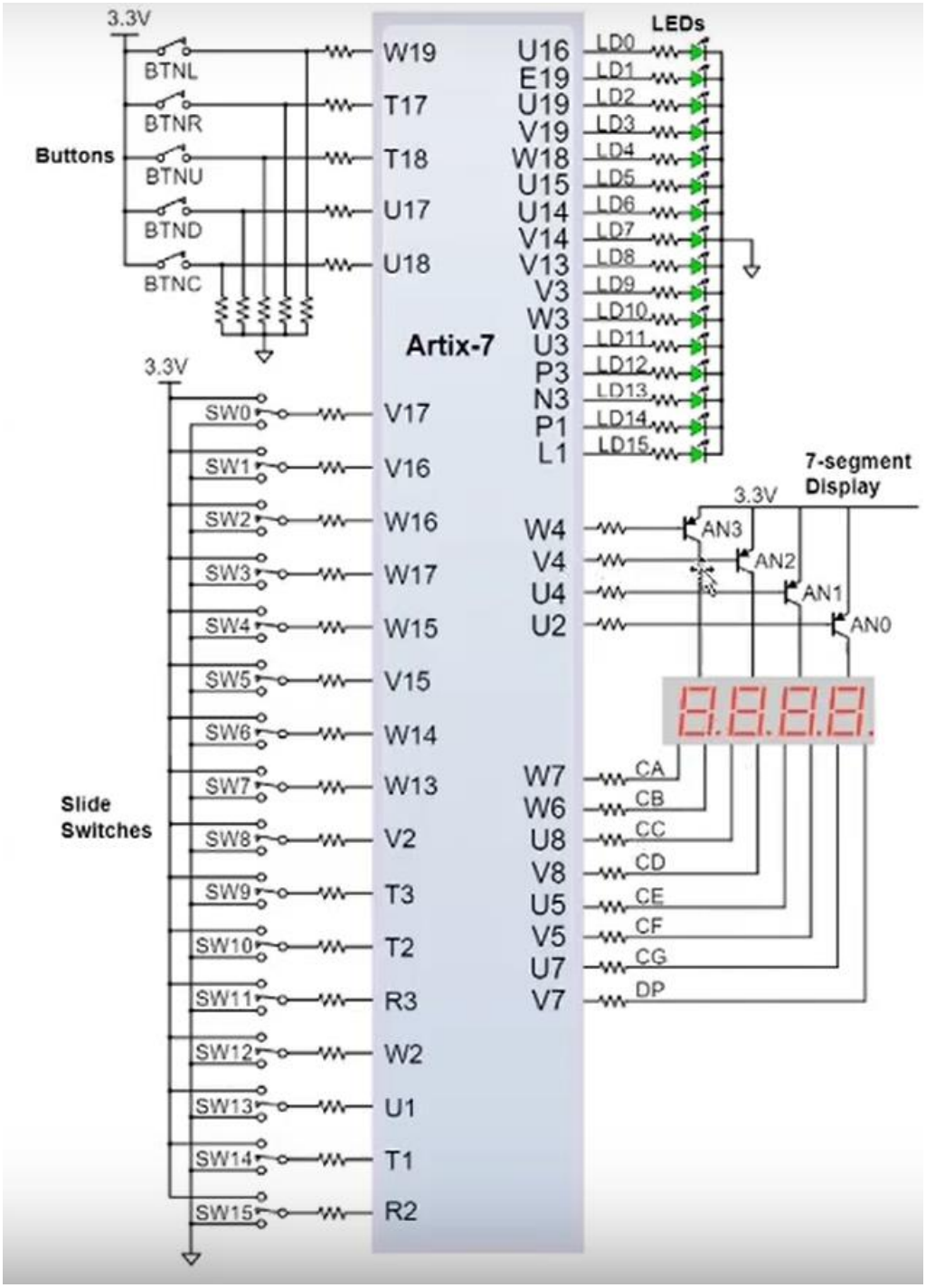

Figure 2 (How does 7 segment work)

Design Specifications:

Persistence of vision is provided on this 7 segment display with a clock frequency of 100 MHz. This watch has two inputs and two outputs. Inputs are reset button and fixed clock frequency (100 MHz). We used the seven segment display as a counter. But this counter displays numbers in hexadecimal system. In this project, a main module was created. This main module uses the clock and driver modules.

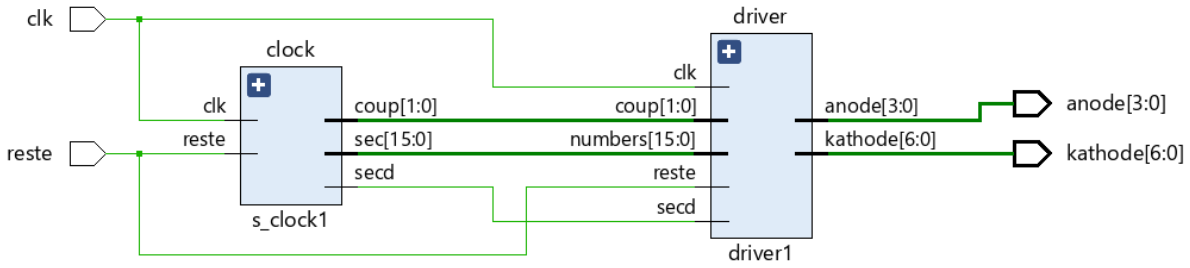

Figure 3 (my rtl)

Results:

Thanks to the features of vhdl, it is possible to test the code and create simulations with testbench. Minor mistakes made while writing the testbench code can affect the result.

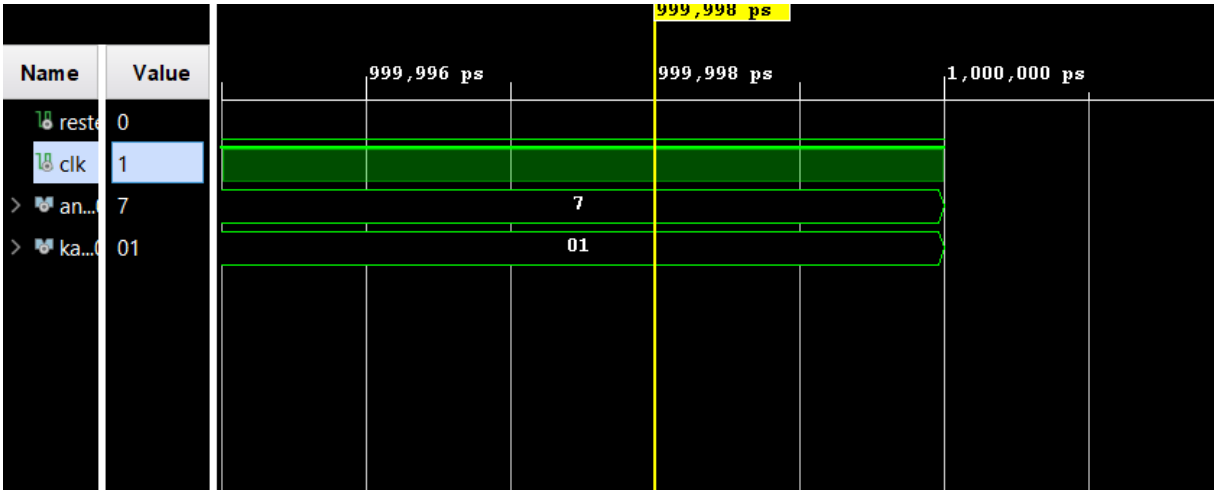

Figure 4 (Simulation for 7 segment chronometer)

In addition, the main module created is implemented in basys3. And it was photographed and shared while it was in reset mode and working.

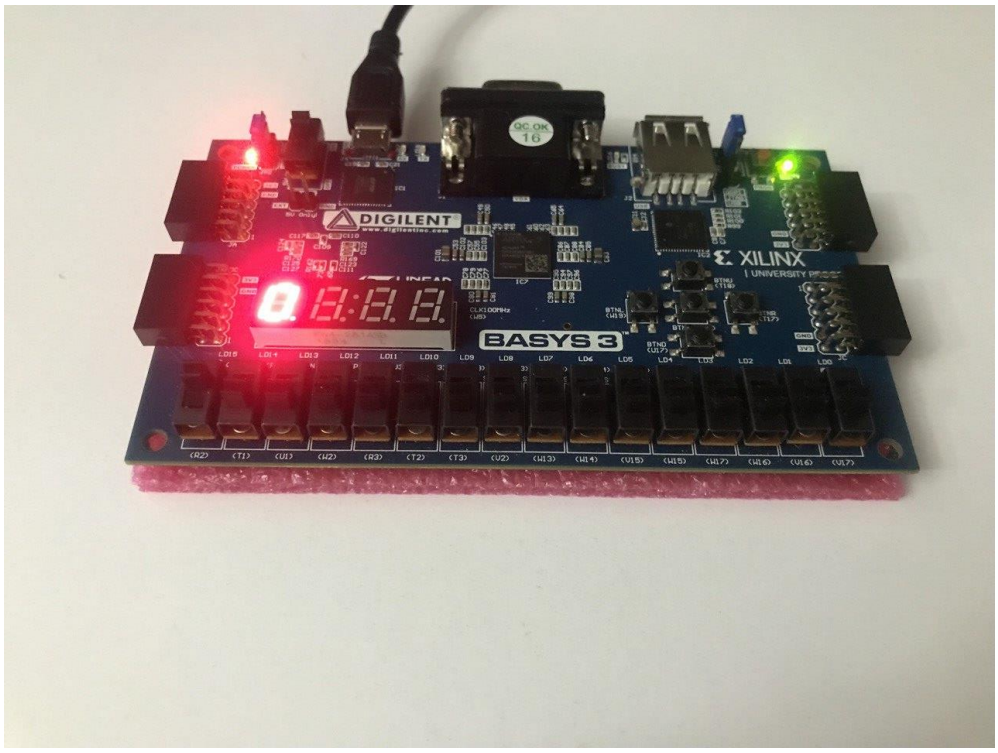

Figure 5 (when reset mod is active)

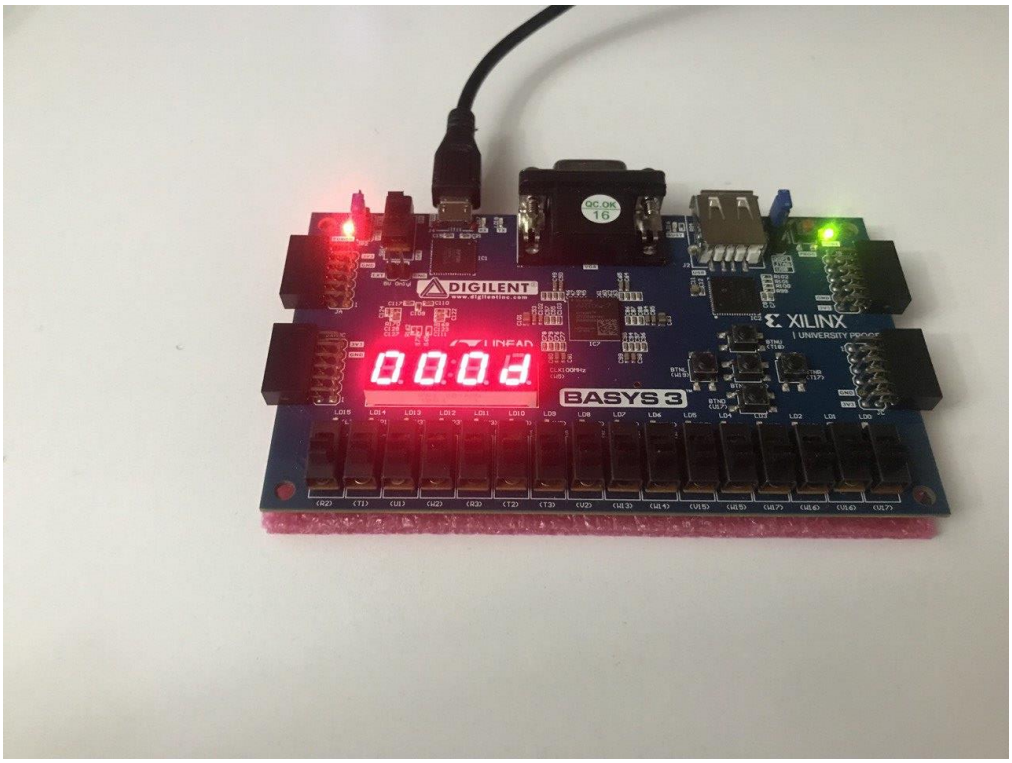

Figure 6 (13 seconds after the reset)

Conclusion:

A hierarchical module has been created in this project. The anode cathode concepts, which play the most important role in the operation of the 7-segment display, are examined. A counter in hexadecimal number system was designed by using sub-modules in a hierarchical order by creating a main module.

Appendices:

- **For main module**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity main1 is
    Port ( clk : in STD_LOGIC;
           reste : in STD_LOGIC;
           anode : out STD_LOGIC_VECTOR (3 downto 0);
           kathode : out STD_LOGIC_VECTOR (6 downto 0));
end main1;

architecture rtl of main1 is
signal coup: std_logic_vector(1 downto 0);
signal secd: std_logic;
```

```
signal numbers: std_logic_vector(15 downto 0);


begin
clock: entity work.s_clock1(rtl_clock)
   PORT MAP(clk => clk, reste => reste, coup => coup, secd => secd, sec => numbers);
driver: entity work.driver1(rtl_driver)
   port map (clk => clk, reste => reste, coup => coup, numbers => numbers, secd => secd,
anode => anode, kathode => kathode);
end rtl;
```

- **For clock**

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


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;


entity s_clock1 is
   Port ( clk : in STD_LOGIC;
          reste : in STD_LOGIC;
          counter : out STD_LOGIC_VECTOR (27 downto 0);
          coup : out STD_LOGIC_VECTOR (1 downto 0);
```

```
        sec : out STD_LOGIC_VECTOR (15 downto 0);
        secd : out std_logic);
end s_clock1;

architecture rtl_clock of s_clock1 is
signal counter_op: std_logic_vector(27 downto 0);
signal s_coun: std_logic_vector(15 downto 0):=(others => '0');

begin
process(clk) begin
   if(reste = '1') then
      counter_op <= (others => '0');
      s_coun <= (others => '0');
    else
      if rising_edge(CLK) then
         counter_op <= counter_op + '1';
         if counter_op =x"5F5E0FF" then
            s_coun <= s_coun+'1';
            counter_op <= (others => '0');
         end if;
      end if;
   end if;
end process;

secd <= '1' when counter_op =x"5F5E0FF" else '0';
coup <= counter_op(19 downto 18);
counter <= counter_op;
```

```
sec <= s_coun;
end rtl_clock;
```

- **For driver**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity driver1 is
    Port ( clk : in STD_LOGIC;
           reste : in STD_LOGIC;
           coup : in STD_LOGIC_VECTOR (1 downto 0);
           numbers : in STD_LOGIC_VECTOR (15 downto 0);
           secd : in STD_LOGIC;
           anode : out STD_LOGIC_VECTOR (3 downto 0);
           kathode : out STD_LOGIC_VECTOR (6 downto 0));
end driver1;

architecture rtl_driver of driver1 is
```

```
    signal led: std_logic_vector(3 downto 0);
    begin
    process (coup) begin
       case coup is
          when "00" => anode <= "0111";
             led <= numbers(15 downto 12);
          when "01" => anode <= "1011";
             led <= numbers(11 downto 8);
          when "10" => anode <= "1101";
             led <= numbers(7 downto 4);
          when "11" => anode <= "1110";
             led <= numbers(3 downto 0);
          when others => anode <= "1111";
       end case;
    end process;
    leds: entity work.decoder(rtl_LED)
       port map(led=>led, kathode => kathode);



    end rtl_driver;
```

- **For decoder**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
```

```
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decoder is
    Port ( led : in STD_LOGIC_VECTOR (3 downto 0);
           kathode : out STD_LOGIC_VECTOR (6 downto 0));
end decoder;

architecture rtl_led of decoder is

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

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

end rtl_led;
```

- **For seven segment display testbench**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity test_bench_main is
end test_bench_main;
architecture beh of test_bench_main is
signal reste,clk: std_logic;
signal anode: std_logic_vector(3 downto 0);
signal kathode: std_logic_vector(6 downto 0);
begin
dut: entity work.main1(rtl)
   port map (clk => clk, reste=>reste,
anode => anode,
kathode=> kathode
);
clock_process :process
```

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

stim_proc: process
begin
reste <= '1';
wait for 20 ns;
reste <= '0';
wait;
end process;

end beh;
```

Reference:

https://github.com/CankutBoraTuncer