Report: Seven-Segment Display

Bilkent University Electrical and Electronics Department

EE102-01 Lab 5

Melih Ege Doğan – 22403329

March 24, 2025

Purpose:

The goal of this lab is to learn how to control multiple seven-segment displays on a Basys 3 board so they can show different numbers at the same time. However, the hardware can only light up one digit at a time. To get around this, we use a trick called “persistence of vision.” This means we switch between digits very quickly, so it looks like all of them are lit up at once.

Questions:

- **What is the internal clock frequency of Basys 3?**
  - Default clock frequency of Basys 3 is 100Mhz.
- **How can you create a slower clock signal from this one?**
  - A slower clock can be created by using a clock divider. If you have an N-bit counter driven by the original clock, you can generate a slower clock by using the most significant bits of that counter.
- **Can you create a clock with any arbitrary frequency lower than that of the internal clock? If not, which frequencies can you create?**
  - You cannot create any arbitrary lower frequency from the Basys 3's 100 MHz clock. Simple clock dividers, using counters, can only produce frequencies that are integer divisions of the original clock. Therefore, you are limited to a specific set of frequencies directly related to the 100 MHz input by whole-number division.

Methodology:

To complete this lab, it was important to understand persistence of vision, which is based on how the human eye reacts to light. This concept is used in the seven-segment display because the LEDs need to turn on and off very quickly to make it look like they are all lit up at the same time. The seven-segment display works by using seven LEDs to form numbers. It has four anodes and seven cathodes to control which LEDs turn on. However, it cannot light up multiple digits at the same time. The first step was to decide on a clock frequency. Since the Basys 3 board has a built-in 100MHz clock, this value was chosen. To make the display work properly, a driver and clock modules were created.

Design Specifications:

For this seven-segment display, two inputs and two outputs were used. The clock runs at 100MHz, which is fast enough for persistence of vision to work. One input is the clock, which keeps the display running, and the other is the reset input, which resets the outputs. The display shows seconds in hexadecimal, allowing the Basys 3 to track time for a longer period. The inputs and outputs are listed in the table below.

```
    CLK: in std_logic
    RST: in std_logic
    anode: out std_logic_vector(3 downto 0)
    cathode: out std_logic_vector(6 downto 0)
```

The seven-segment display module was designed in a modular way. The "topmodule.vhd" file is the main module that uses both the clock and display modules. The inputs and outputs of these modules are listed below.

- clock.vhd

```
CLK: in std_logic
RST: in std_logic
counter: out std_logic_vector(27 downto 0)
second: out std_logic_vector(15 downto 0)
ds: out std_logic_vector (1 downto 0)
is_sec: out std_logic
```

- display.vhd

```
    CLK : in std_logic
    RST : in std_logic
    ds : in std_logic_vector(1 downto 0)
    number: in std_logic_vector(15 downto 0)
    is_sec: in std_logic
    anode : out std_logic_vector(3 downto 0)
    cathode: out std_logic_vector(6 downto 0)
```

Results:

The suggested seven-segment display design, along with its schematics, simulations, and photos of the implementation on the Basys 3, can be found. The code for the modules is included in the appendices.

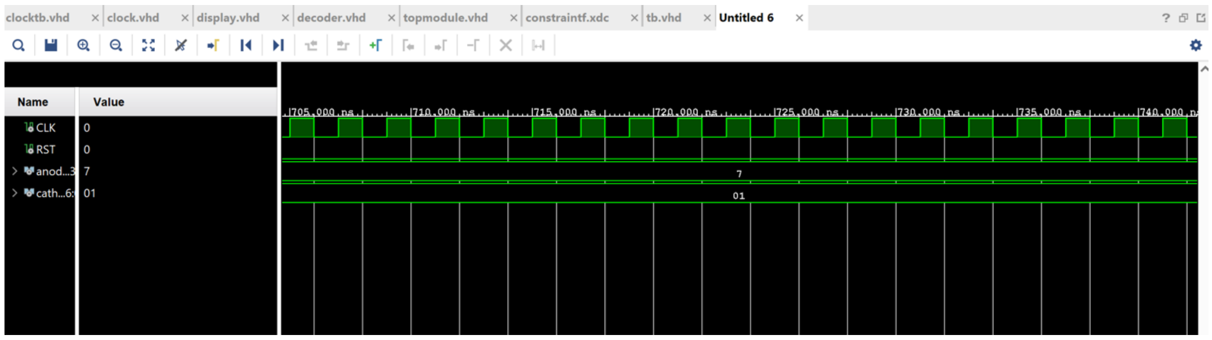


Figure 1: Testbench of “topmodule.vhd”.

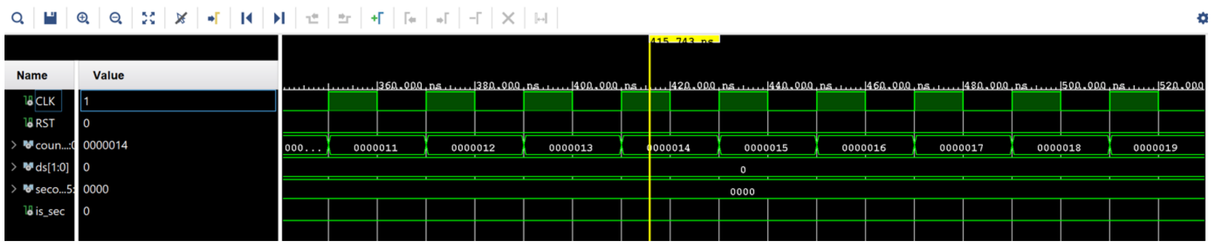


Figure 2: Testbench for “clock.vhd”.

The next section includes the schematic of the seven-segment display and the top module’s schematic.

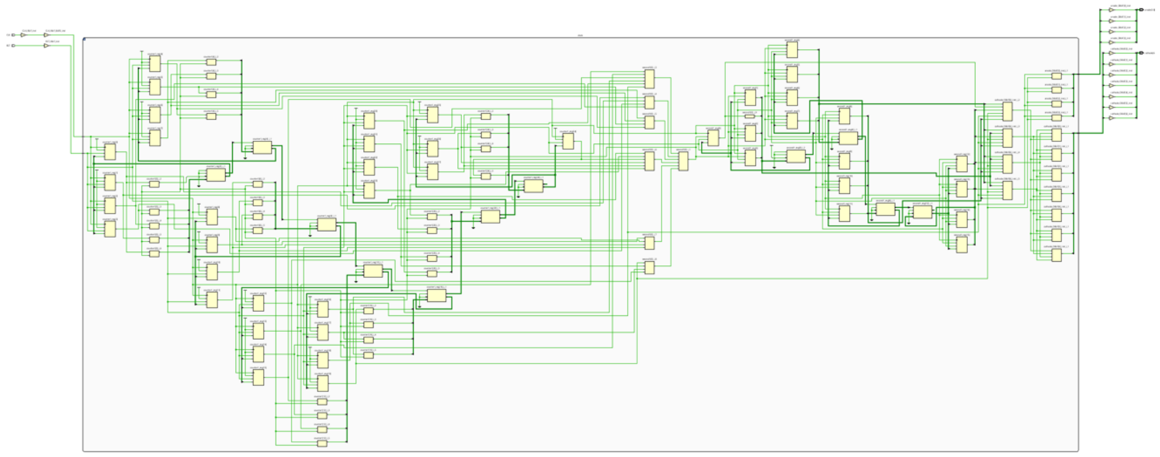


Figure 3: Synthesized Schematic.

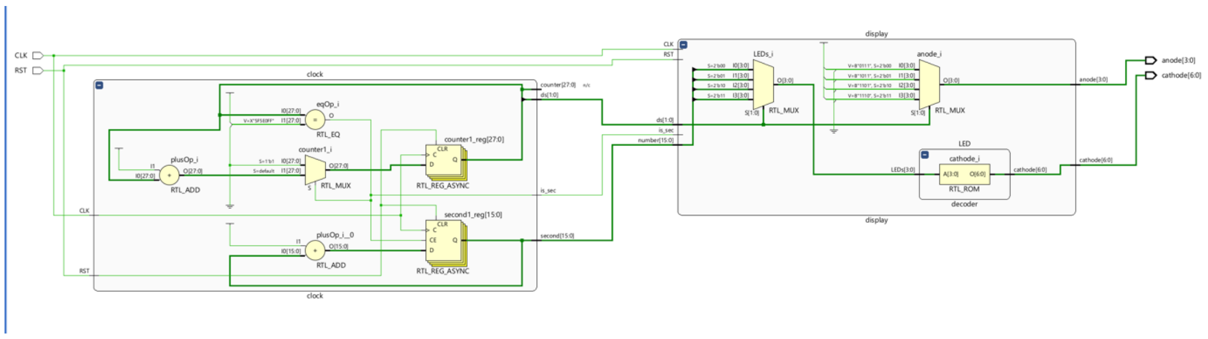
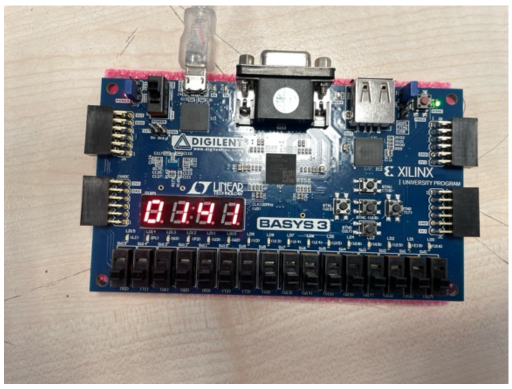
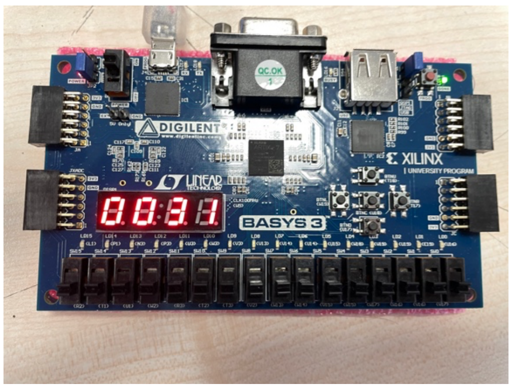


Figure 4: RTL Schematic.

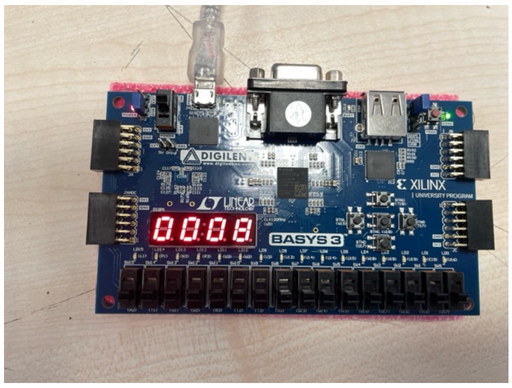
To demonstrate the design on the Basys 3 board, the following images have been selected to show some of the moments. It is important to remember that this design uses base 16.



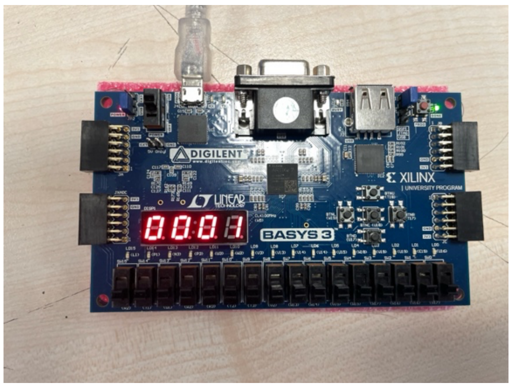
a) After 321 seconds.



b) After 49 seconds.



a) After 8 seconds.



b) After 1 seconds.

Figure 5: Pictures of working Basys Second Counter.

Conclusion:

This lab successfully demonstrated how to display multiple digits on a Basys 3's seven-segment display, despite the hardware limitation of only lighting one digit at a time. We achieved this by implementing a time-multiplexed display driver that exploits the persistence of vision. A clock divider, created using a counter, generated the necessary timing signals to rapidly switch between displaying each hexadecimal digit of the second counter. The testbench simulations and the physical implementation on the Basys 3 board confirmed the correct functionality of the design.

Appendices:

Code 1: topmodule.vhd

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



entity topmodule is
  Port (
    CLK: in std_logic;
    RST: in std_logic;
    anode: out std_logic_vector(3 downto 0);
    cathode: out std_logic_vector(6 downto 0)
 );
end topmodule;

architecture Behavioral of topmodule is

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

begin

clock: entity work.clock(Behavioral) PORT MAP(CLK => CLK, RST => RST, ds => ds,
is_sec => is_sec, second => number);
display: entity work.display(Behavioral) PORT MAP(CLK => CLK, RST => RST, ds => ds,
number => number, is_sec => is_sec, anode => anode, cathode => cathode);
end Behavioral;
```

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);
  second: out std_logic_vector(15 downto 0);
  ds: out std_logic_vector (1 downto 0);
  is_sec: out std_logic
  );
end clock;

architecture Behavioral of clock is
    signal counter1: std_logic_vector(27 downto 0);
    signal second1: std_logic_vector(15 downto 0) := (others => '0');

begin
process(CLK)
begin
    if(RST = '1') then
    counter1 <= (others => '0'); second1 <= (others => '0');

    elsif(rising_edge(CLK)) then
        counter1 <= counter1 +1;
        if(counter1 = x"5F5E0FF") then
            second1 <= second1 +1; counter1 <= (others => '0');
        end if;
    end if;
end process;


is_sec <= '1' when counter1 =x"5F5E0FF" else '0';
ds <= counter1(19 downto 18);
counter <= counter1;
second <= second1;


end Behavioral;
```

Code 3: display.vhd

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

entity display is
  Port (
    CLK : in std_logic;
    RST : in std_logic;
    ds : in std_logic_vector(1 downto 0);
    number: in std_logic_vector(15 downto 0);
    is_sec: in std_logic;
    anode : out std_logic_vector(3 downto 0);
    cathode: out std_logic_vector(6 downto 0)

  );
```

```
end display;

architecture Behavioral of display is
    SIGNAL LEDs: std_logic_vector(3 downto 0);
begin
process(ds)
begin
    case ds is
        when "00" => anode <= "0111"; LEDs <= number(15 downto 12);
        when "01" => anode <= "1011"; LEDs <= number(11 downto 8);
        when "10" => anode <= "1101"; LEDs <= number(7 downto 4);
        when "11" => anode <= "1110"; LEDs <= number(3 downto 0);
        when others => anode <= "1111";
    end case;

end process;

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

end Behavioral;
```

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 Behavioral of decoder is

begin
process(LEDs) is
    begin
         case LEDs is
            when "0000" => cathode <= "0000001";
            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 <= "0001000";
            when "1011" => cathode <= "1100000";
            when "1100" => cathode <= "0110001";
            when "1101" => cathode <= "1000010";
            when "1110" => cathode <= "0110000";
            when others => cathode <= "0111000";
         end case;
    end process;

end Behavioral;
```

Code 5: tb.vhd

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

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

architecture Behavioral of tb 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.topmodule(Behavioral)
PORT MAP (CLK => CLK, RST=>RST, anode=>anode, cathode=>cathode);

clock_process : process begin
CLK <= '0';
wait for 1 ns;
CLK <= '1';
wait for 1 ns;
end process;
stim_proc: process begin
RST <= '1';
wait for 2 ns;
RST <= '0';
wait;
end process;

end Behavioral;
```

Code 6: clocktb.vhd

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

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

architecture Behavioral of clocktb is
signal CLK: std_logic;
signal RST: std_logic;
signal counter: std_logic_vector(27 downto 0);
signal ds: std_logic_vector(1 downto 0);
signal second: std_logic_vector(15 downto 0);
signal is_sec: std_logic;
begin

dut: entity work.clock
PORT MAP( CLK => CLK, RST => RST, counter => counter, ds =>ds, second => second,
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 Behavioral;
```

References:

- https://github.com/SemihAkkoc/EEE102
- https://en.wikipedia.org/wiki/Persistence_of_vision
- https://en.wikipedia.org/wiki/Seven-segment_display
- https://nandland.com/project-5-seven-segment-display/