Tuna Şahin
22201730

# EEE 102-2 Lab Report 5 - Seven Segment Counter

## Questions to Answers:

1) What is the internal clock frequency of Basys 3?
- The internal clock frequency of Basys3 is 100 MHz.

2) How can you create a slower clock signal from this one?
- By changing another binary value every “n” rising-edge of the internal clock signal or overflowing another binary value and finding its period.

3) Can you create a clock with any arbitrary frequency lower than that of the internal clock? If not, which frequencies can you create?
- Getting another frequency by overflowing another binary value, we can only obtain frequencies that are (10ns*2^n). However, by increasing another binary value every “n” rising-edge of the clock we by using another counter variable we can get every integer multiple of 10ns.

## Experimental Work:

### Introduction:

In this lab, we used the internal clock of Basys3 and its seven segment display to make a counter that increments every second. We used the persistence of vision effect to make the different digits appear like they are on at the same time. The counter counts using decimal number system.

### Methodology:

The digits of the seven segment display cannot turn on at the same time as each other. To display 4 digits simultaneously we make use of an effect called “persistance of vision”. We turn on and off all the digits one by one very quickly. For the effect to occur the period of the lights must be between 1-16 ms. In this case the period was roughly 10.5 ms. Each of these lights turn on for 1/4th of the period. I took the most two significant bits through a decoder to decide which digit will light up. I then ran the binary values corresponding to the of the digits through multiple multiplexeres to get the equivalent seven segment cathode binary values.

### Circuit:

### Clock:

The Clock Circuit takes the clk signal (integrated clock signal of basys3) and the reset signal (connected to left button) and outputs a 4-bit binary signal that corresponds to an anode. The reset signal sets the counter to 0 whenever it is on. If the reset is off, then every 10ns a 20-bit signal is incremented by 1. The most two significant digits of the 20-bit signal corresponds to the 4-bit anode output through a decoder.

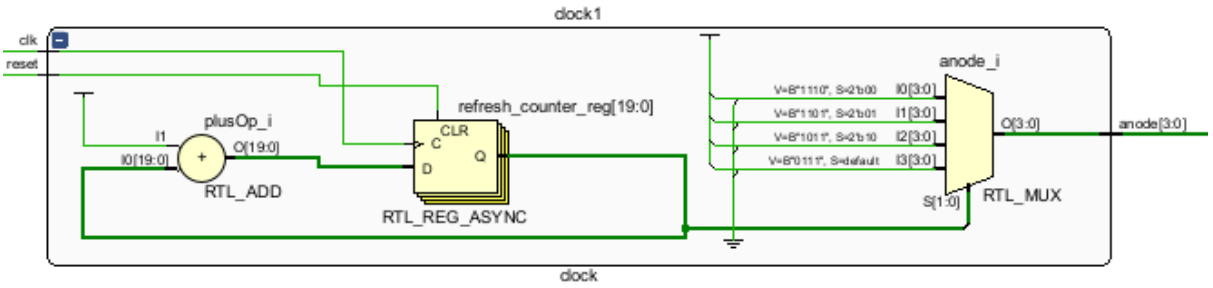


Figure 1.1: Clock Circuit Schematic

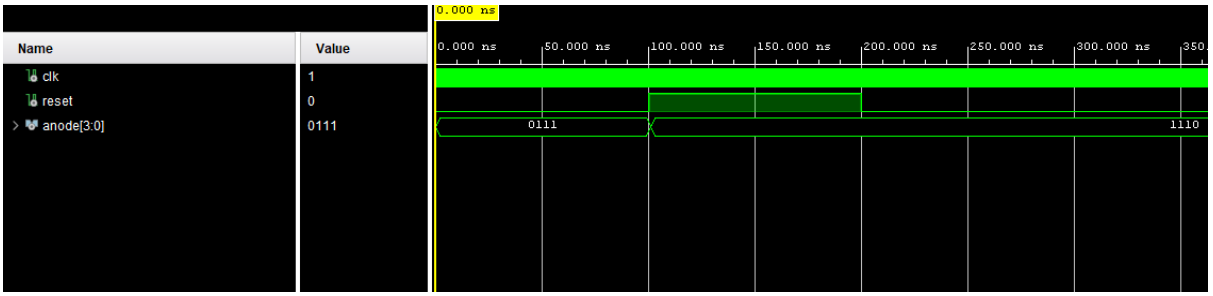


Figure 1.2: Clock Circuit Simulation at start (reset is visible)

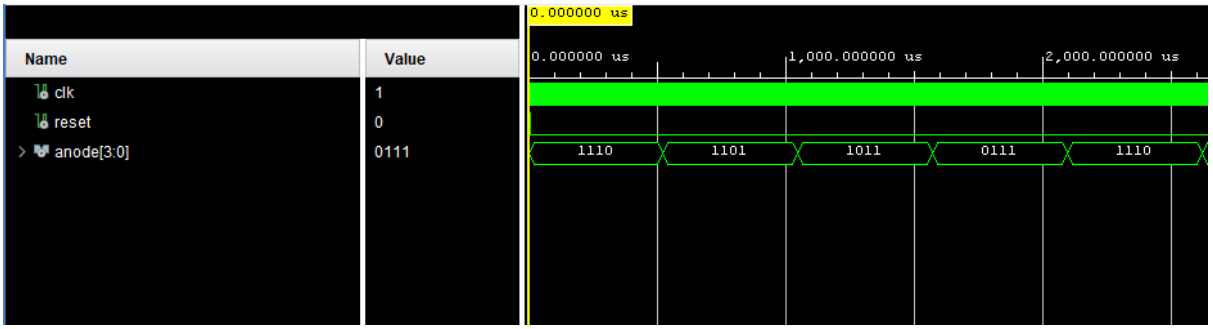


Figure 1.3: Clock Circuit Simulation

## binary2cathode Circuit:

A 4-bit signal is present for every digit. However, these signals need to be converted to their 7-bit seven segment display equivalent. A rom is used to accomplish this. A rom (read only memory) stores the 7-bit values and the respective values are selected by the 4-bit input signals.

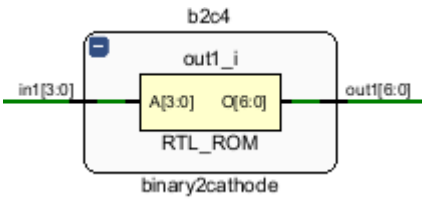


Figure 2.1: binary2cathode Circuit Schematic

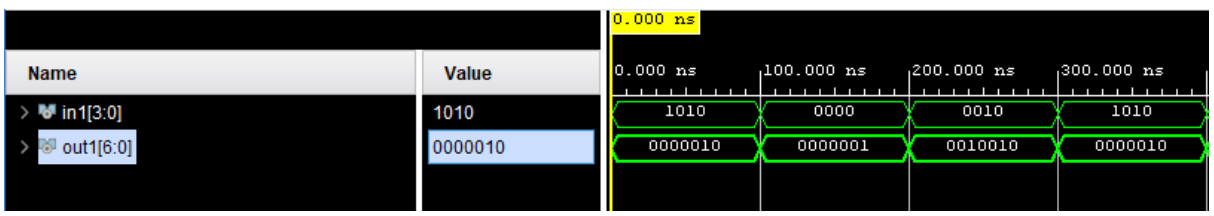


Figure 2.2: binary2cathode Circuit Simulation

## Counter Circuit:

Contrary to the popular approach where a hexadecimal counter is being used, this circuit utililises a decimal counter. This is why the schematic is quite larger and different than most circuits. This is accomplished by 5 binary variables. A 28-bit and 4 4-bit variables to be exact. The 28-bit signal resets once it reaches “5f5e0ff” in hexadecimal. Every reset of the 28-bit signal, one of the 4-bit variables increase by 1. Each 4-bit signal resets once it reaches “1001” and increments the next 4-bit signal. These 4-bit signals each correspond to the values sent to the binary2cathode circuit for each digit. The reset signal again sets every signal to 0.

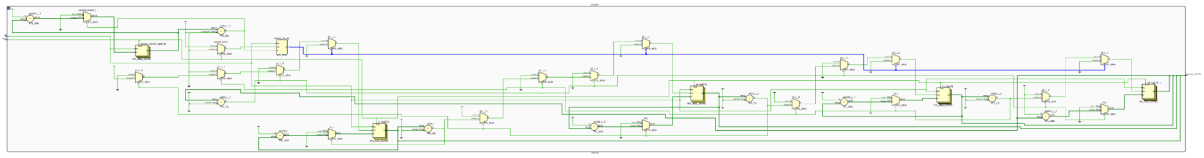

Figure 3.1: Counter Schematic (HD version included in the end)

## Main:

The only logic that happens in main circuit deciding which 7-bit signal corresponding to the active digit will be outputted to the cathodes. This is done through a multiplexer. Also the 4 4-bit signals corresponding to the digit values are displayed on the LEDs for debugging purposes.

## Conclusion:

In this lab we utilised the internal clock of BASYS3 FGPA board. I also used the buttons on the BASYS3 while programming the reset input. The main focus of this lab was to understand how the seven segment display worked. The seven segment display consists of four anodes and 7 cathodes. Each anode controls the digit that will be active while cathodes control the led that will light up. Since two or more of the digits of the seven segment display cannot be active at the same time, we used “persistance of vision” effect. This effect suggests that out eyes perceive flickering with a period of 1-16 ms as a permanent stimuli. Taking advantage of this effect we cycled through the digits of the seven segment display to make it appear as if all the digits are on at the same time. By the end of the lab, I realised and learned how and where to use the seven segment display on the BASYS3.

# VHDL Code and Pictures

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

entity main is
  Port (clk,reset : in std_logic;
        anode : out std_logic_vector(3 downto 0);
        led : out std_logic_vector(15 downto 0);
        val: out std_logic_vector(6 downto 0));
end main;

architecture Behavioral of main is
    component clock is
        Port (clk,reset: in std_logic;
              anode : out std_logic_vector(3 downto 0));
    end component;

    component counter is
         Port (seconds_out : out std_logic_vector(15 downto 0);
               clk,reset : in std_logic);
    end component;

    component binary2cathode is
         Port (in1 : in std_logic_vector(3 downto 0);
               out1: out std_logic_vector(6 downto 0));
    end component;

    signal seconds : std_logic_vector(15 downto 0);
    signal led1,led2,led3,led4 : std_logic_vector(6 downto 0);
    signal anode_t : std_logic_vector(3 downto 0);
begin
    clock1 : clock port map     (clk => clk, reset => reset, anode => anode_t);
    counter1 : counter port map (clk => clk, reset => reset, seconds_out => seconds);
    b2c1 : binary2cathode port map(in1 => seconds(15 downto 12),out1 => led1);
    b2c2 : binary2cathode port map(in1 => seconds(11 downto 8),out1 => led2);
    b2c3 : binary2cathode port map(in1 => seconds(7 downto 4),out1 => led3);
    b2c4 : binary2cathode port map(in1 => seconds(3 downto 0),out1 => led4);

    process(anode_t)
    begin
       case anode_t is
         when "1110" => val <= led4;
         when "1101" => val <= led3;
         when "1011" => val <= led2;
         when others => val <= led1;
       end case;
    end process;
    anode <= anode_t;
    led <= seconds;
end Behavioral;
```

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

entity clock is
  Port (clk,reset: in std_logic;
        anode : out std_logic_vector(3 downto 0));
end clock;

architecture Behavioral of clock is
   signal refresh_counter : std_logic_vector(19 downto 0);
begin
   process(clk,reset)
   begin
      if reset = '1' then
         refresh_counter <= (others => '0');
      elsif rising_edge(clk) then
         refresh_counter <= refresh_counter + 1;
      end if;

      case refresh_counter(19 downto 18) is
        when "00" =>   anode <= "1110";
        when "01" =>   anode <= "1101";
        when "10" =>   anode <= "1011";
        when others => anode <= "0111";
      end case;
   end process;
end Behavioral;
```

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

entity counter is
  Port (seconds_out : out std_logic_vector(15 downto 0);
        clk,reset : in std_logic);
end counter;

architecture Behavioral of counter is
   signal second_counter : std_logic_vector(27 downto 0);
   signal second_inc: std_logic;
   signal s1,s2,s3,s4 : std_logic_vector(3 downto 0);
begin
   process(clk,reset)
   begin
      if reset = '1' then
         second_counter <= (others => '0');
      elsif rising_edge(clk) then
         if second_counter = x"5f5e0ff" then
            second_counter <= (others => '0');
            second_inc <= '1';
         else
            second_inc <= '0';
            second_counter <= second_counter + 1;
         end if;
      end if;

      if reset = '1' then
         s1 <= (others => '0');
         s2 <= (others => '0');
         s3 <= (others => '0');
         s4 <= (others => '0');
      elsif rising_edge(clk) then
         if second_inc = '1' then
            if s4 = "1001" then
               s4 <= "0000";
            elsif s3 = "1001" then
               s3 <= "0000";
               s4 <= s4 + 1;
            elsif s2 = "1001" then
               s2 <= "0000";
               s3 <= s3 + 1;
            elsif s1 = "1001" then
               s1 <= "0000";
               s2 <= s2 + 1;
            else
               s1 <= s1 + 1;
            end if;
         end if;
       end if;
   end process;
   seconds_out <= s4 & s3 & s2 & s1;
end Behavioral;
```

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

entity binary2cathode is
  Port (in1 : in std_logic_vector(3 downto 0);
        out1: out std_logic_vector(6 downto 0));
end binary2cathode;

architecture Behavioral of binary2cathode is

begin
    process(in1)
    begin
       case in1 is
       when "0000" => out1 <= "0000001"; -- "0"
       when "0001" => out1 <= "1001111"; -- "1"
       when "0010" => out1 <= "0010010"; -- "2"
       when "0011" => out1 <= "0000110"; -- "3"
       when "0100" => out1 <= "1001100"; -- "4"
       when "0101" => out1 <= "0100100"; -- "5"
       when "0110" => out1 <= "0100000"; -- "6"
       when "0111" => out1 <= "0001111"; -- "7"
       when "1000" => out1 <= "0000000"; -- "8"
       when "1001" => out1 <= "0000100"; -- "9"
       when "1010" => out1 <= "0000010"; -- a
       when "1011" => out1 <= "1100000"; -- b
       when "1100" => out1 <= "0110001"; -- C
       when "1101" => out1 <= "1000010"; -- d
       when "1110" => out1 <= "0110000"; -- E
       when others => out1 <= "0111000"; -- F
       end case;
    end process;

end Behavioral;
```

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
 set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN W19 [get_ports reset]
      set_property IOSTANDARD LVCMOS33 [get_ports reset]
#seven-segment LED display
set_property PACKAGE_PIN W7 [get_ports {val[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[6]}]
set_property PACKAGE_PIN W6 [get_ports {val[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[5]}]
set_property PACKAGE_PIN U8 [get_ports {val[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[4]}]
set_property PACKAGE_PIN V8 [get_ports {val[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[3]}]
set_property PACKAGE_PIN U5 [get_ports {val[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[2]}]
set_property PACKAGE_PIN V5 [get_ports {val[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[1]}]
set_property PACKAGE_PIN U7 [get_ports {val[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {val[0]}]

set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anode[3]}]

set_property PACKAGE_PIN U16 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property PACKAGE_PIN U19 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property PACKAGE_PIN V19 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property PACKAGE_PIN W18 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property PACKAGE_PIN U15 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property PACKAGE_PIN U14 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property PACKAGE_PIN V14 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property PACKAGE_PIN V13 [get_ports {led[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
set_property PACKAGE_PIN V3 [get_ports {led[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
set_property PACKAGE_PIN W3 [get_ports {led[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
set_property PACKAGE_PIN U3 [get_ports {led[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
set_property PACKAGE_PIN P3 [get_ports {led[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]

set_property PACKAGE_PIN N3 [get_ports {led[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
set_property PACKAGE_PIN P1 [get_ports {led[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
set_property PACKAGE_PIN L1 [get_ports {led[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]
```

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

entity clock_testbench is
end clock_testbench;

architecture Behavioral of clock_testbench is
   component clock is
       Port (clk,reset: in std_logic;
            anode : out std_logic_vector(3 downto 0));
   end component;

   signal clk,reset : std_logic;
   signal anode : std_logic_vector(3 downto 0);
begin
   clock1 : clock port map (clk => clk, reset => reset, anode => anode);
   process
   begin
      clk <= '1';
      wait for 1ns;
      clk <= '0';
      wait for 1ns;
   end process;

   reset_proc : process
   begin
      reset <= '0';
      wait for 100ns;
      reset <= '1';
      wait for 100ns;
      reset <= '0';
      wait;
    end process;

end Behavioral;
```

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

entity b2c_test is
end b2c_test;

architecture Behavioral of b2c_test is
   component binary2cathode is
      Port (in1 : in std_logic_vector(3 downto 0);
           out1: out std_logic_vector(6 downto 0));
   end component;

   signal in1 : std_logic_vector(3 downto 0);
   signal out1: std_logic_vector(6 downto 0);
begin
   b2c1 : binary2cathode port map(in1 => in1,out1 => out1);
   process
   begin
      in1 <= "1010";
      wait for 100ns;
      in1 <= "0000";
      wait for 100ns;
      in1 <= "0010";
      wait for 100ns;
   end process;

end Behavioral;
```

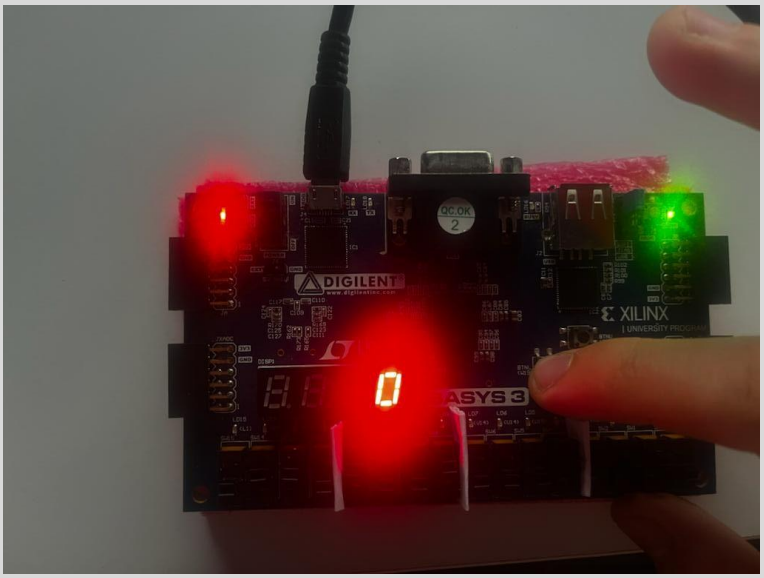


Figure 4.1: BASYS when reset is ‘1’

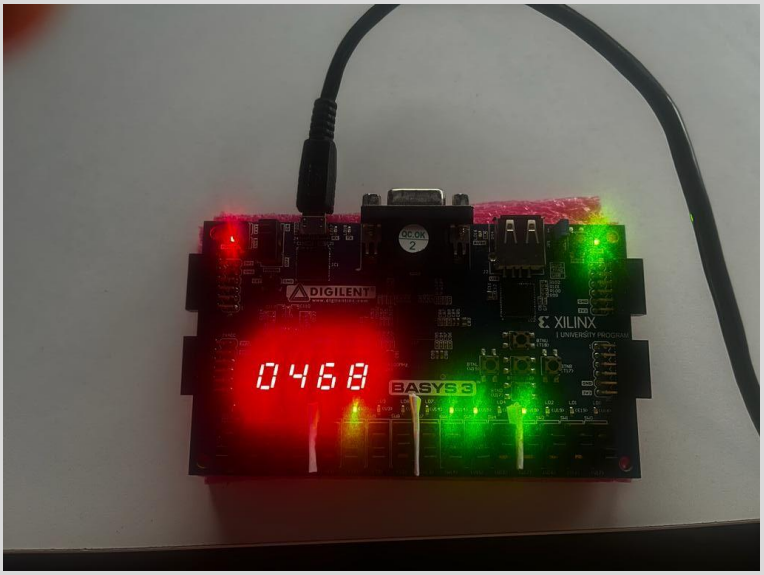


Figure 4.2: BASYS after 468 seconds (counter is decimal)

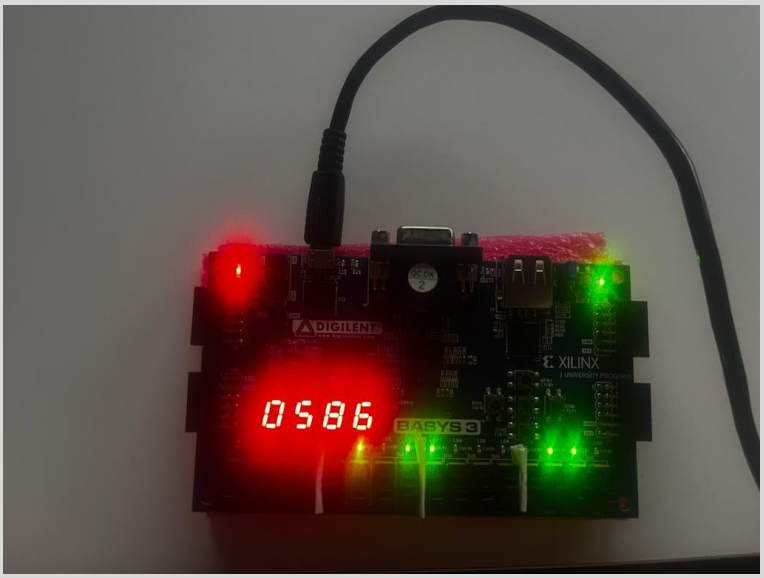


Figure 4.3: BASYS after 586 seconds (counter is decimal)