EEE102-02 Lab 5 Report:

Seven-Segment Display

Kaan Ermertcan - 22202823

30.10.2023

Purpose:

In this lab, we will design a circuit that displays multiple digits on the seven-segment display simultaneously and implement it on a Basys 3 board. We will use the persistence of vision effect when displaying different digits on Basys3's seven-segment display.

Design Specifications:

My design is supposed to take an arbitrary 12-bit unsigned binary number as the input, convert it to decimal and display the decimal representation of that number on the seven-segment display. 12 switches starting from the right on the Basys3 are used for the input. The rightmost switch is taken as the least significant bit. A 1KHz clock is created to run and update the seven-segment display. 1KHz is enough to create a persistence of vision effect.

Methodology:

First, the clock signal at 1KHz should be acquired from the internal 100MHz clock of Basys3 by dividing the internal clock by 10^5 in the clock divider module (clock.vhd). This is done by counting the rising edges of the internal clock and changing the created clock signal when required number of (10^5) internal clock pulses are counted. Using the same method, internal clock frequency can be divided by any integer. Basys3 also contains clock management tiles (CMTs), each consisting of one MMCM and one PLL. With the use of those, the internal clock can be divided to fractional numbers. Thus, arbitrary clock speeds can be achieved, albeit within some accuracy. As MMCM's support division by 0.125, clock frequencies up to 800MHz can be achieved (AMD, 2018 & Gisselquist, 2019).

Second, the 12-bit binary input is converted to decimal and separated into its digits in the converter module (converter.vhd). Functions from numeric_std library are used to achieve this (converted binary to decimal using to_integer function, isolated individual digits using mod function and arithmetic operators). This module outputs a 16-bit vector where its each 4-bit group represents a decimal digit.

Finally, the top module sevensegment (sevensegment.vhd) combines the clock divider and converter and drives the seven-segment display at the clock speed from clock divider according to the output from the converter module (internal clock clockin is connected to the input of clock divider, switch_input from the switches is connected to the input of the converter and output of clock divider is taken as the clock signal, output of the converter is taken as sseg signal). Multiplexers are created to switch digits and segments. Each digit (anode) is cycled continuously, changing every millisecond(1/1kHz) and segments(cathodes) are switched according to the selected digit at the same frequency. Because Basys3 has transistors connected to each anode, anodes are active low instead of active high. Thus, to light up a segment, outputs connected to both its anode and cathode must be asserted low. Two vectors, one for anodes and one for cathodes, are used as the output and connected to seven-segment display (sseganode: 4-bit: corresponding to digits from left to right (MSB to LSB), ssegcathode: 8-bit: corresponding to segments DP, G, …., A from MSB to LSB). Dot point is included but always asserted high (never lit).

A testbench (sevensegment_test.vhd) is created for simulation purposes, which contains two processes. One process simulates the clock input and the other is for simulating different switch inputs.

Results:

RTL schematics of my designed modules can be found below.

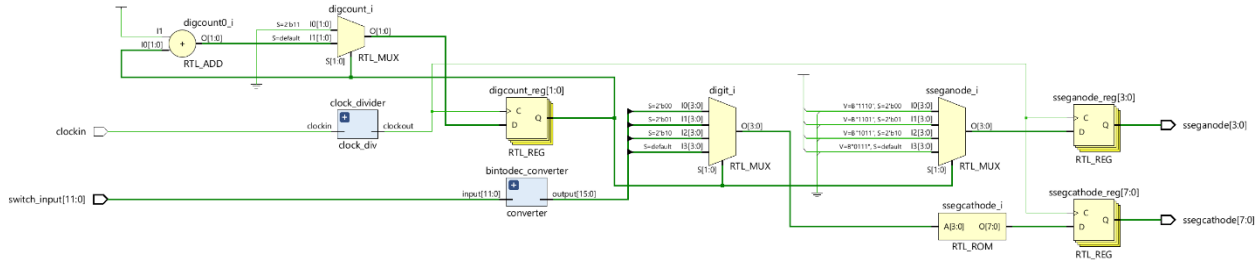

Figure 1.1: RTL of sevensegment.

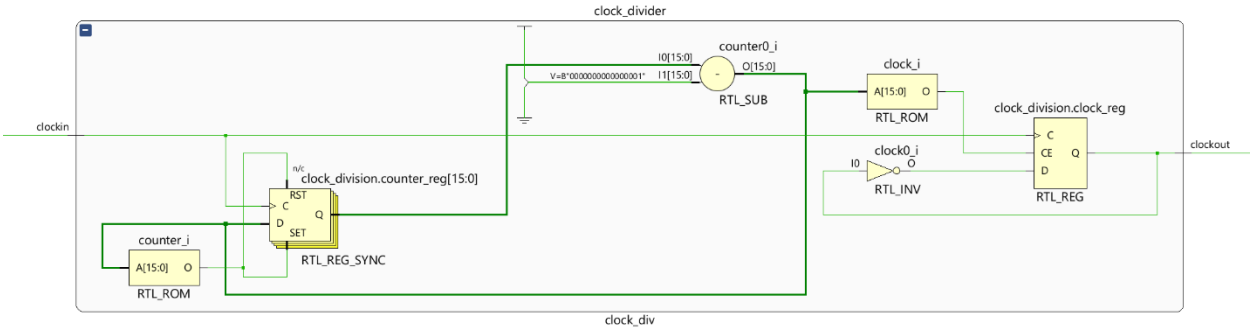

Figure 1.2: RTL of clock_div.

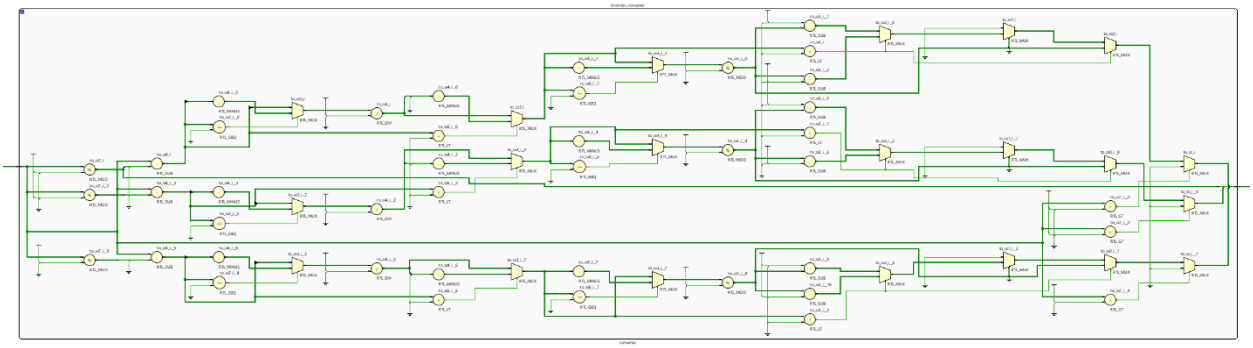

Figure 1.3: RTL of converter.

As can be seen from the schematics, registers are created for keeping the variables, components shaped as yellow circles are created for arithmetic operations, and muxes are created for switching.

Simulation results of some sample inputs can be found below.

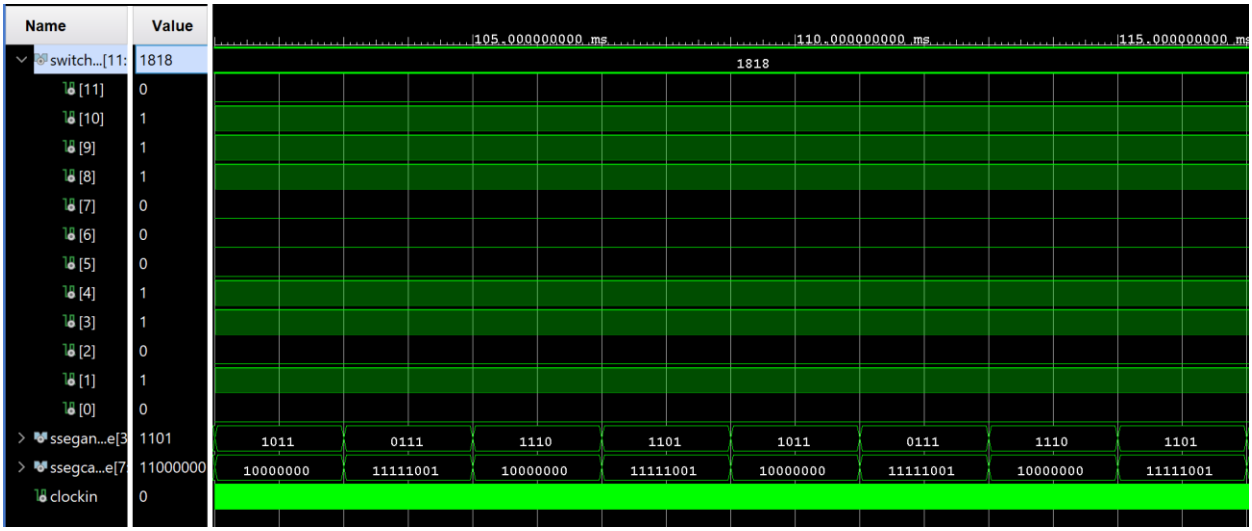

Figure 2.1: Input $011100011010_2 = 1818_{10}$.

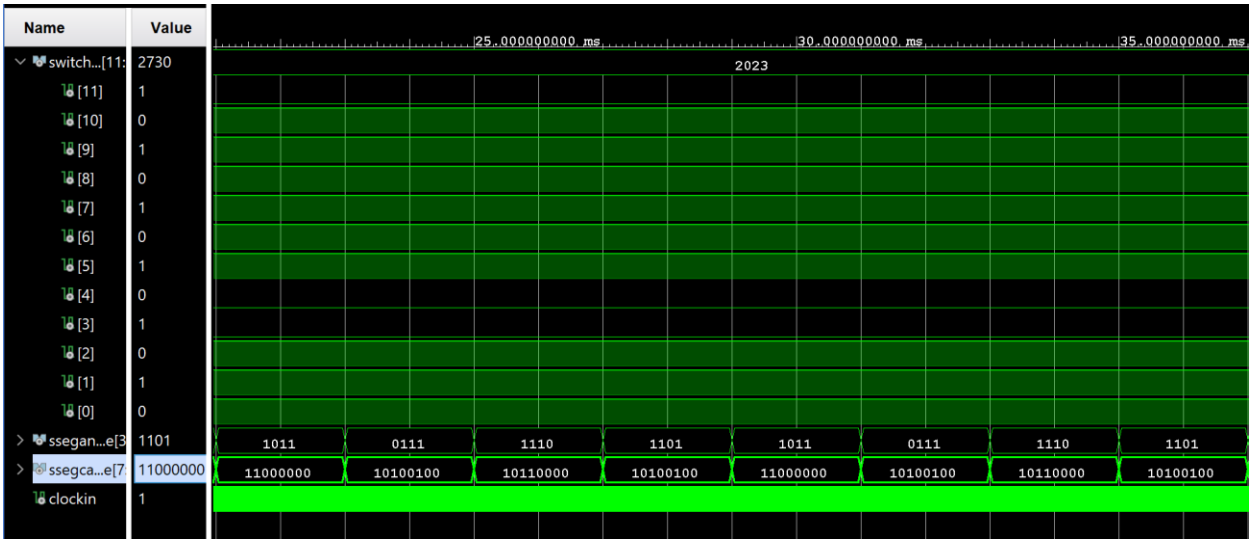

Figure 2.2: Input $011111100111_2 = 2023_{10}$.

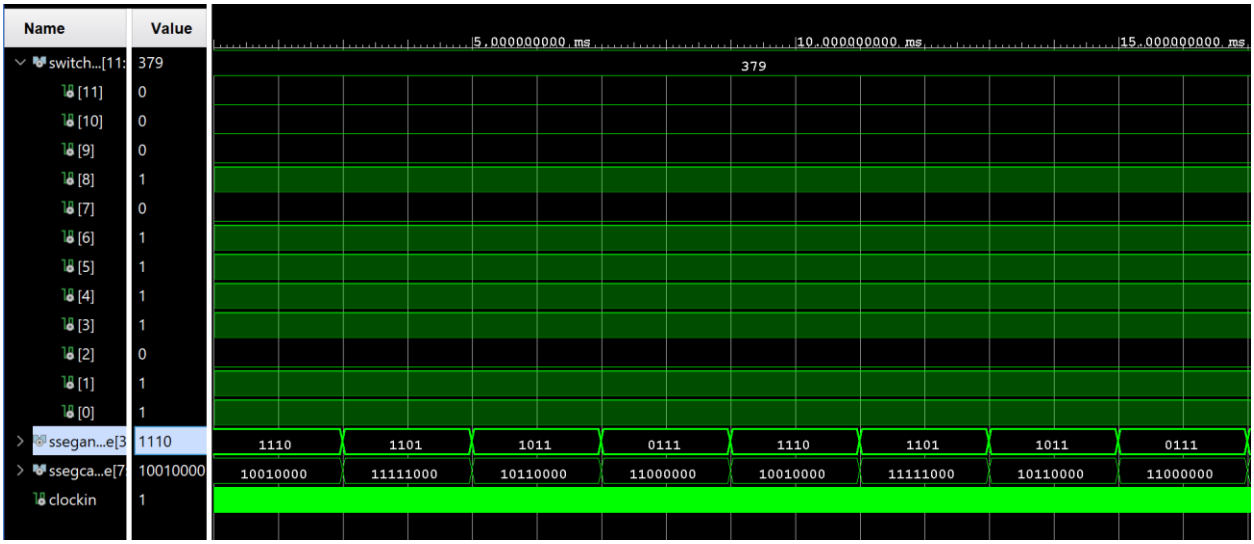

Figure 2.1: Input $000101111011_2 = 379_{10}$.

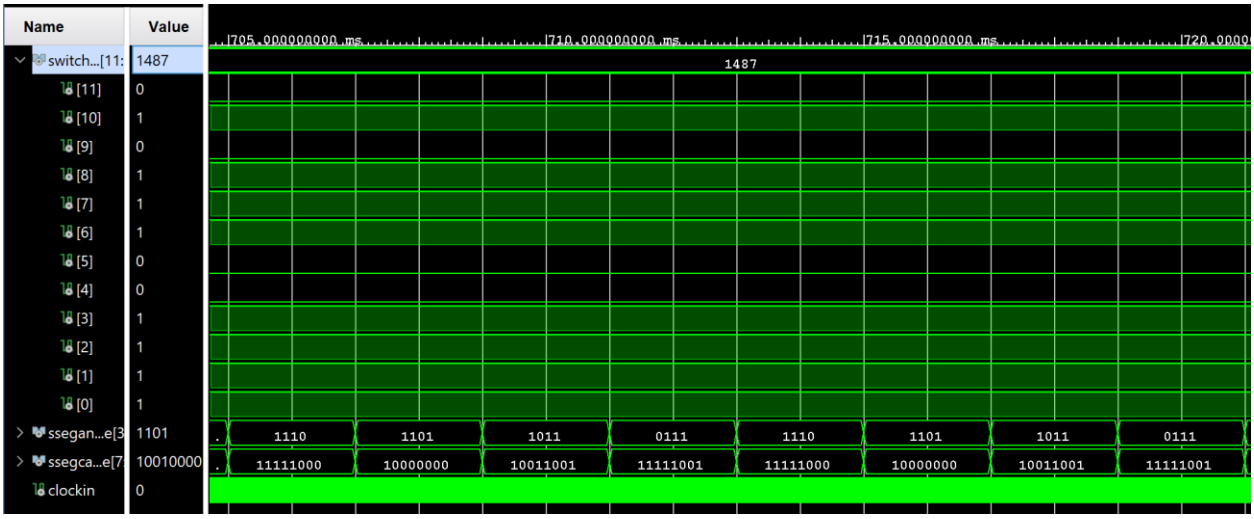

Figure 2.1: Input $010111001111_2 = 1487_{10}$.

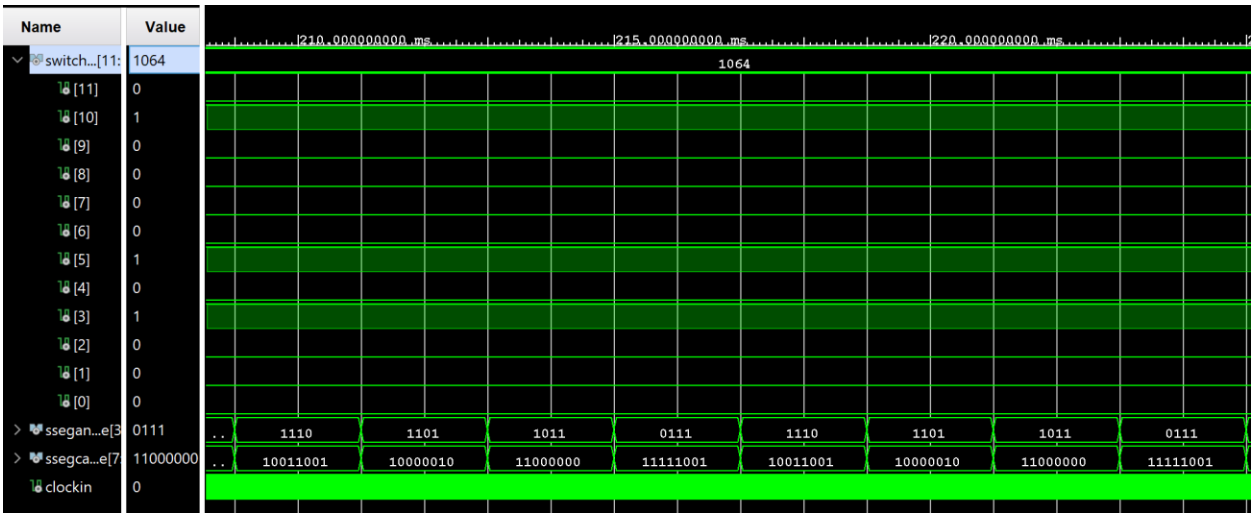

Figure 2.1: Input $010000101000_2 = 1064_{10}$.

As can be seen, sseganode and ssegcathode outputs are asserted as expected depending on time and switch_input.

Sample photos of the Basys3 can be found below.

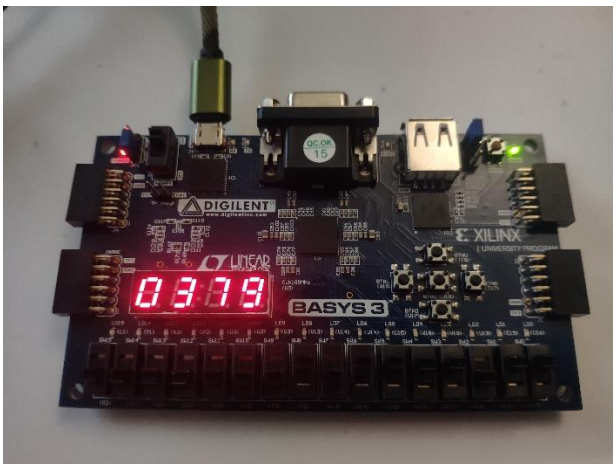

Figure 3.1: Input $000101111011_2 = 379_{10}$.

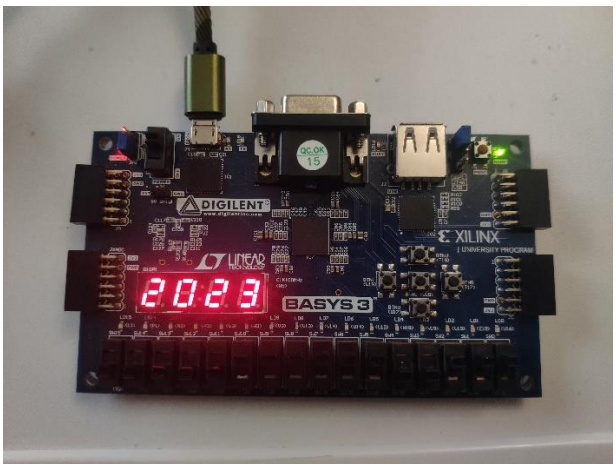

Figure 3.2: Input $011111100111_2 = 2023_{10}$.

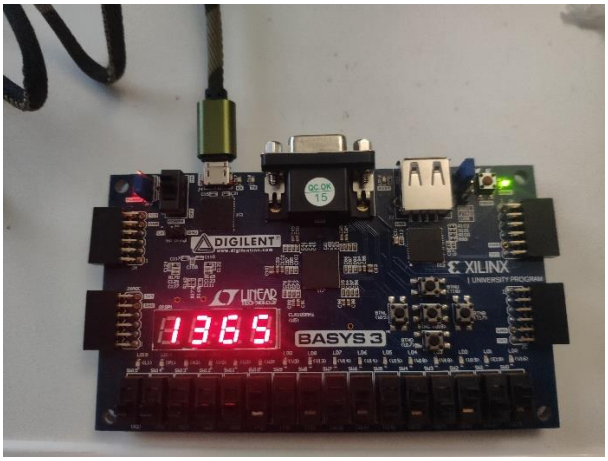

Figure 3.3: Input $010101010101_2 = 1365_{10.}$

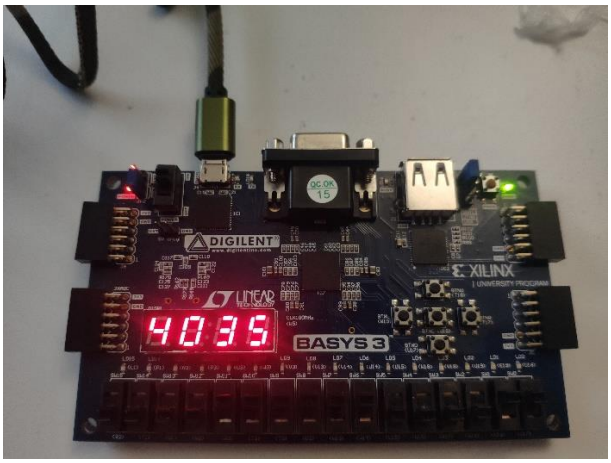

Figure 3.4: Input $111111000011_2 = 4035_{10.}$

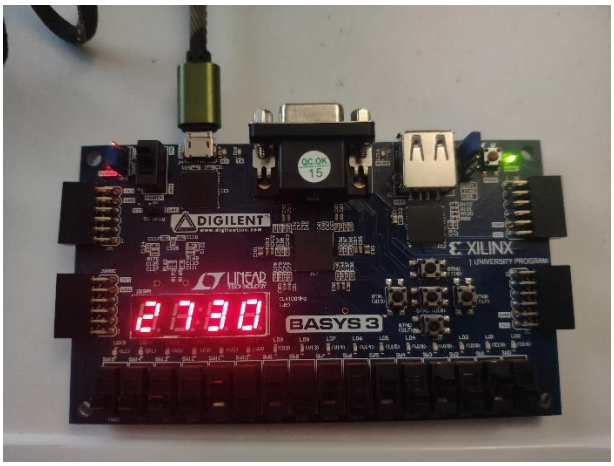

Figure 3.5: Input $101010101010_2 = 2730_{10.}$

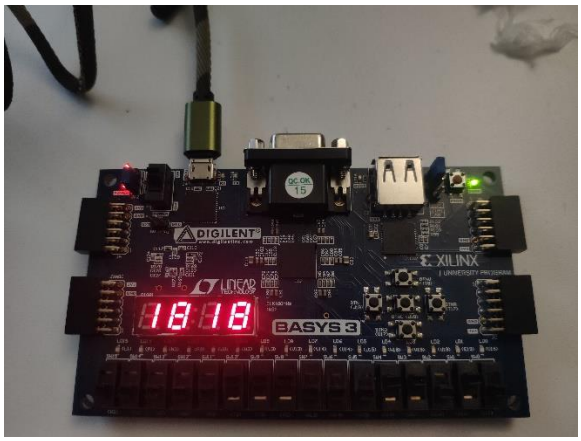

Figure 3.6: Input $011100011010_2 = 1818_{10.}$

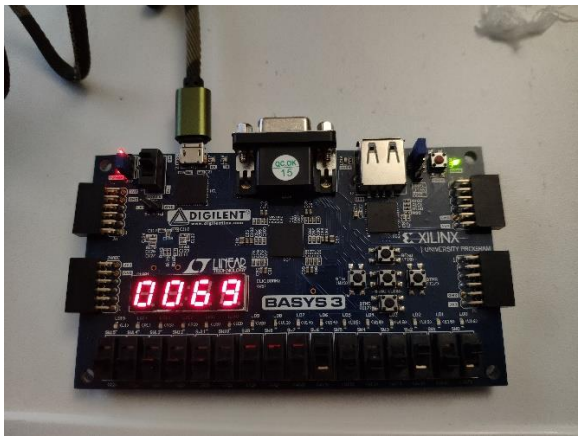

Figure 3.7: Input $000001000101_2 = 69_{10.}$

As can be seen, binary numbers entered through the switches are converted to decimal and displayed correctly on the seven-segment display. Persistence of vision is also achieved.

Conclusion:

In this lab, I have successfully implemented a design that takes a 12-bit binary input and displays its decimal representation on the seven-segment display. I learned how to use the clock signal and create a clock with a different frequency than the internal clock. This was my first design that is time dependent (includes sequential logic) but I managed to implement it.

References:

AMD. (2018, July 30). *7 Series FPGAs Clocking Resources User Guide (UG472)*. AMD Adaptive Computing Documentation Portal. https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking

Gisselquist, D. (2019, June 28). Breaking all the rules to create an arbitrary clock signal. https://zipcpu.com/blog/2019/06/28/genclk.html

Appendices:

Code 1: sevensegment.vhd

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

entity sevensegment is
   Port ( switch_input : in std_logic_vector (11 downto 0);
        sseganode : out STD_LOGIC_VECTOR (3 downto 0);
        ssegcathode : out STD_LOGIC_VECTOR (7 downto 0);
        clockin : in std_logic);
end sevensegment;

architecture Behavioral of sevensegment is
signal clock : std_logic := '0';
signal sseg : STD_LOGIC_VECTOR (15 downto 0);
component clock_div is
   Port (clockin : in std_logic;
      clockout : out STD_LOGIC);
end component;
component converter is
Port (input : in STD_LOGIC_VECTOR (11 downto 0);
    output: out STD_LOGIC_VECTOR (15 downto 0));
end component;
begin
bintodec_converter: converter port map (input => switch_input, output => sseg);
```

```
clock_divider: clock_div port map (clockout => clock, clockin => clockin);
process(clock)
variable digit : std_logic_vector (3 downto 0) := "0000" ;              -- digit to be displayed
variable digcount : integer range 0 to 3 := 0;       -- to keep track of the anode that is asserted low
begin
if rising_edge(clock) then
case digcount is              -- switches anodes and corresponding digit
when 0 => digit := sseg(3 downto 0); sseganode <= "1110";
when 1 => digit := sseg(7 downto 4); sseganode <= "1101";
when 2 => digit := sseg(11 downto 8); sseganode <= "1011";
when others => digit := sseg(15 downto 12); sseganode <= "0111";
end case;
case digit is                 -- switches cathodes according to the digit
when "0000" => ssegcathode <= "11000000";
when "0001" => ssegcathode <= "11111001";
when "0010" => ssegcathode <= "10100100";
when "0011" => ssegcathode <= "10110000";
when "0100" => ssegcathode <= "10011001";
when "0101" => ssegcathode <= "10010010";
when "0110" => ssegcathode <= "10000010";
when "0111" => ssegcathode <= "11111000";
when "1000" => ssegcathode <= "10000000";
when "1001" => ssegcathode <= "10010000";
when others => ssegcathode <= "11111111";
end case;
if digcount = 3 then
digcount := 0;
else
digcount := digcount + 1;
end if;
end if;
end process;
end Behavioral;
```

Code 2: clock.vhd

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

entity clock_div is
   Port ( clockin : in STD_LOGIC;
          clockout : out STD_LOGIC);
```

```
end clock_div;

architecture Behavioral of clock_div is
begin
clock_division: process(clockin)
variable counter : integer range 0 to 50000 := 50000;
variable clock : std_logic := '0';
begin
   if rising_edge(clockin) then
   counter := counter-1;
      if counter = 0 then
      clock := not clock;
      counter := 50000;
      end if;
   end if;
   clockout <= clock;
end process;
end Behavioral;
```

Code 3: converter.vhd

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

entity converter is
   Port (input : in STD_LOGIC_VECTOR (11 downto 0);
         output: out STD_LOGIC_VECTOR (15 downto 0));
end converter;

architecture Behavioral of converter is
begin
converter: process(input)
variable decimal : integer range 0 to 4095 := 0;
variable to_ss : unsigned (15 downto 0);
begin
decimal := to_integer(signed(input));
to_ss := (others => '0');
to_ss(3 downto 0) := to_unsigned(decimal mod 10, 4);                    -- Isolating least significant digit
if decimal > 9 then
```

```
to_ss(7 downto 4) := to_unsigned(((decimal - (decimal mod 10))/10) mod 10, 4);      -- Isolating second least significant digit
end if;
if decimal > 99 then
to_ss(11 downto 8) := to_unsigned(((decimal - (decimal mod 100))/100) mod 10, 4);   -- Isolating second most significant digit
end if;
if decimal > 999 then
to_ss(15 downto 12) := to_unsigned(((decimal - (decimal mod 1000))/1000) mod 10, 4);-- Isolating most significant digit
end if;
output <= std_logic_vector(to_ss);
end process;
end Behavioral;
```

Code 4: sevensegment_test.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.math_real.all;

entity sevensegment_test is
end sevensegment_test;

architecture Behavioral of sevensegment_test is
component sevensegment is
Port(     switch_input : in std_logic_vector (11 downto 0);
        sseganode : out STD_LOGIC_VECTOR (3 downto 0);
        ssegcathode : out STD_LOGIC_VECTOR (7 downto 0);
        clockin : in std_logic);
end component;
signal switch_input : std_logic_vector (11 downto 0);
signal sseganode : STD_LOGIC_VECTOR (3 downto 0);
signal ssegcathode : STD_LOGIC_VECTOR (7 downto 0);
signal clockin : std_logic := '0';
begin
uut: sevensegment port map(clockin => clockin, ssegcathode => ssegcathode, sseganode => sseganode, switch_input => switch_input);

process
```

```
constant clock_period : time := 10ns;
begin
clockin <= '0';
wait for clock_period/2;
clockin <= '1';
wait for clock_period/2;
end process;

process
begin
switch_input <= "000101111011";
wait for 20000000ns;
switch_input <= "011111100111";
wait for 20000000ns;
switch_input <= "010101010101";
wait for 20000000ns;
switch_input <= "101010101010";
wait for 20000000ns;
switch_input <= "111111000011";
wait for 20000000ns;
switch_input <= "011100011010";
wait for 20000000ns;
wait;
end process;
end Behavioral;
```