Emre Akdemir

22302606

EE102-02

# EE-102 LAB 5 REPORT: SEVEN-SEGMENT DISPLAY

26/03/2025

## Purpose

The aim of this lab was to create a seven-segment display on BASYS3. Using the LED section of the BASYS3 board, counting process in hexadecimal system was shown on the LED screen by displaying each segment simultaneously and the effect of "persistence of vision" has investigated.

## Questions

**What is the internal clock frequency of Basys 3?**

The internal clock frequency of the Basys 3 FPGA board is 100 MHz.

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

To create a slower clock signal, you can use a clock divider, which counts clock cycles and toggles an output signal at a desired rate.

**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 an arbitrary frequency due to integer division constraints. You can generate frequencies of the form: $f = \frac{100MHz}{N}$, where $N$ is a positive integer. Also, the frequencies that created can be maximum 100Mhz, not higher than that.

## Methodology

The goal of the design is to create “persistence of vision” effect using LED segments. To show this effect, counting process has been done in hexadecimal system and reflected on the LEDs. To count in hexadecimal system, a decoder which takes inputs from the counter and returns output as the binary equivalents of segments for each cathode. The numbers starting from 0 to 15 in decimal system, turns into numbers 7-bit binary numbers and these numbers are called anode. A multiplexer is used to control the 4 LED cathode, and I used active-low inputs for 4 cathode segments. The four segments cannot be powered at the same time, but since the human eye cannot perceive the emitted light rays from the object after a certain frequency, if the these cathodes are powered in high frequency (e.g internal frequency of the clock: 100 MHz), an optic illusion, which is the visual effect of all the cathodes working at the same time, can be created.

## Design Specifications

The Basys 3 board operates with a fixed 100 MHz internal clock and includes three main inputs and outputs: clk, a fixed clock signal; reset, which resets the counting process; anode (7-bit), controlling the segments of the 7-segment display; and cathode (4-bit), determining the active digit. The top module consists of three key components: the Clock Segment Module, the Multiplexer Module, and the LED Register Module. The Clock Segment Module receives the clk and reset signals, using a 28-bit counter (clk_counter1) that increments on every clock cycle. When clk_counter1 reaches 100,000,000, it resets and increases a separate binary counter by one every second. The 18th and 19th bits of clk_counter1 act as a 2-bit counter (phase_counter), which is used in the Multiplexer Module to cycle through the cathode output at a high enough frequency to maintain the persistence of vision effect. Meanwhile, the sec_en signal updates once per second, providing a stable reference for time tracking. The Multiplexer Module ensures that only one 7-segment digit is active at a time, controlled by phase_counter. Finally, the LED Register Module (Decoder) takes the binary counter value, which increments every second, and converts it into its hexadecimal equivalent, splitting the 16-bit value into four-digit sections to properly drive the anode output for display.

## Results

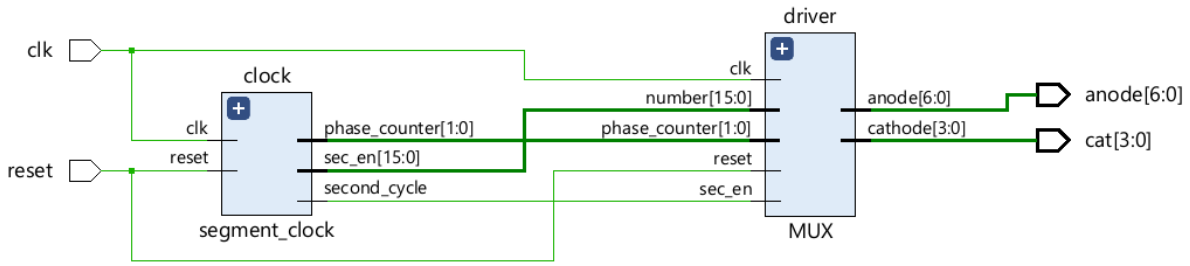


Figure 1: RTL Schematics for the Top Module

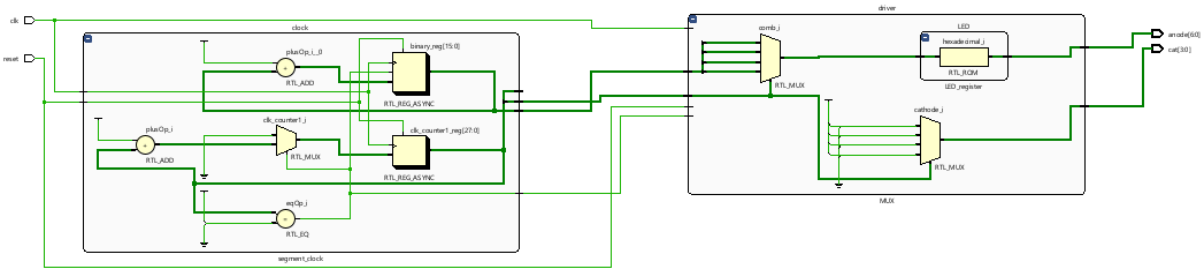

Figure 2: RTL Schematics for the Top Module (Expanded)

The benchmark test sets clock “1” and “0” at each 10 ns, and the reset input is set once as “1” to be tested if it works properly. Results are as expected.

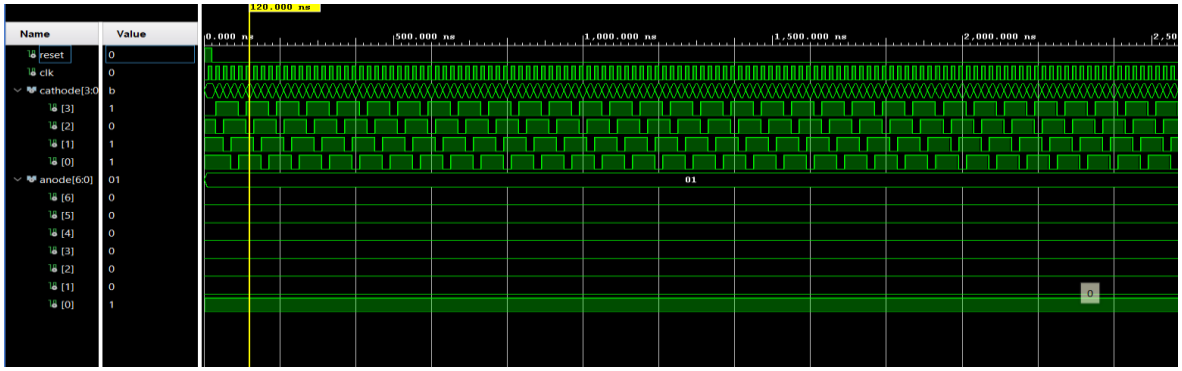


Figure 3: Simulation results of the Top Module

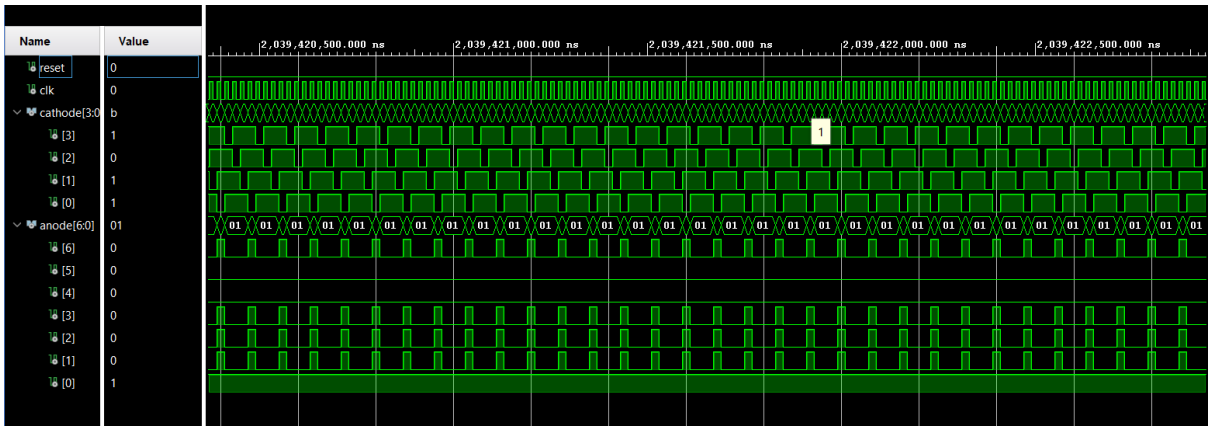


Figure 4: Simulation results of the Top Module

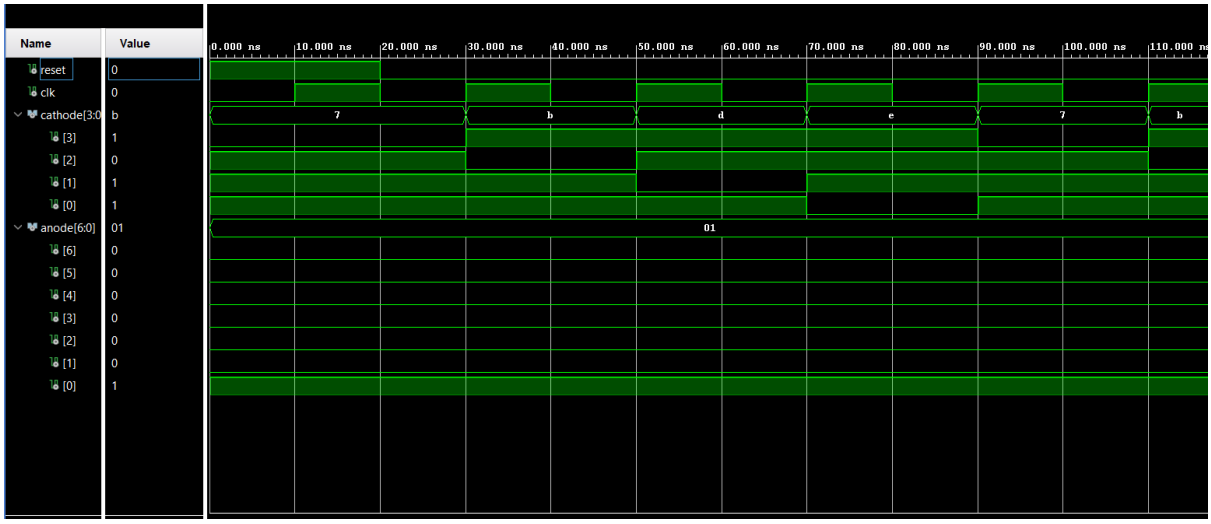


Figure 5: Simulation results of the Top Module (Note that Clock Frequency and the response of Cathode is as expected)

After modifying the constraint file according to the inputs and outputs of the top module, I implemented the design and embedded on Bays3 Board.

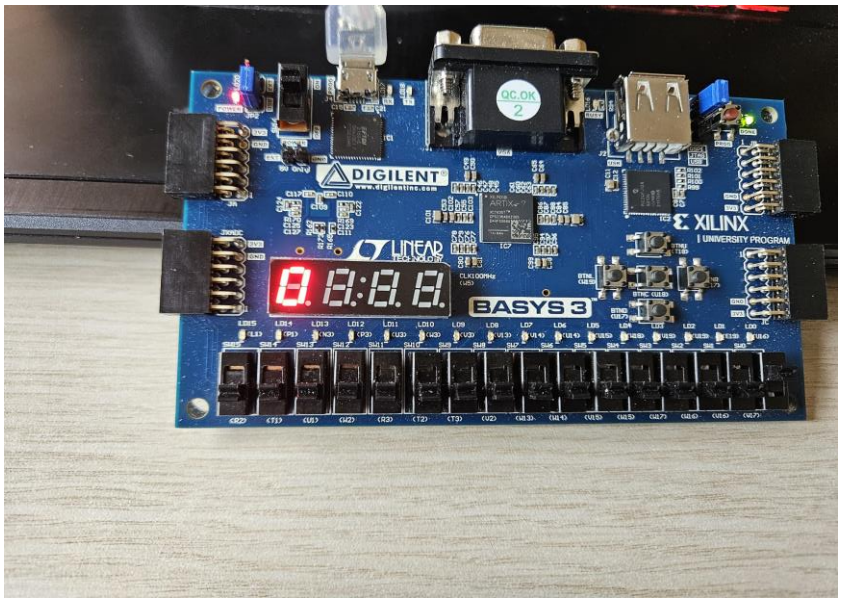


Figure 5: Resetted when the first switch is on

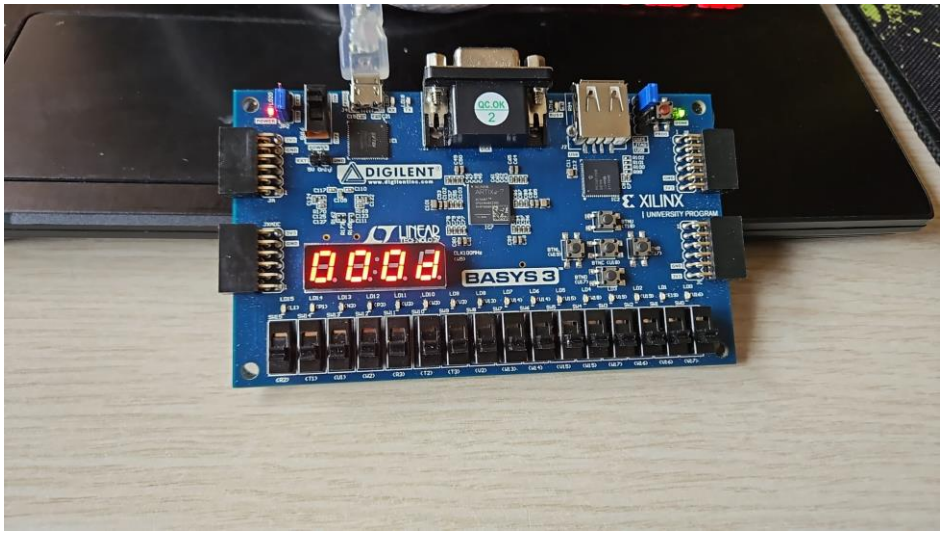


Figure 6: “000d” after 13 seconds

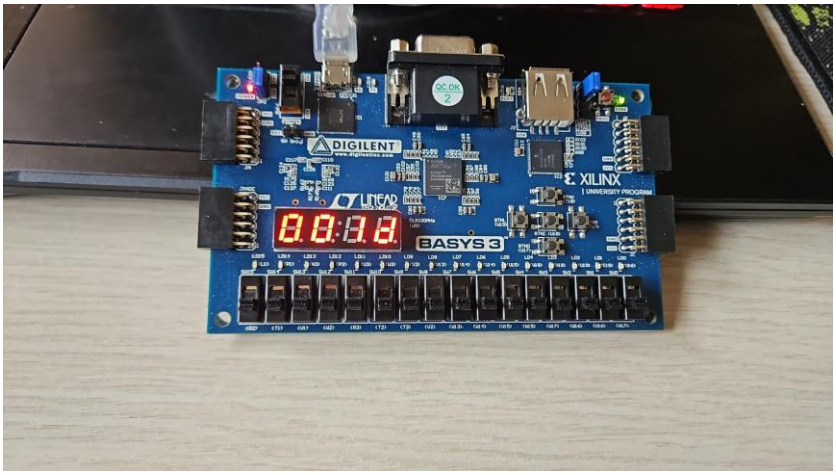


Figure 7: “001d” after 29 seconds

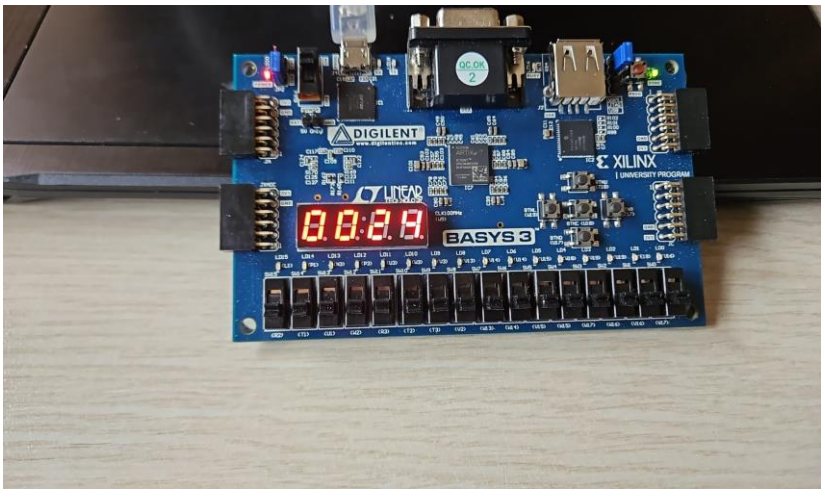

Figure 8: “0024” after 36 seconds

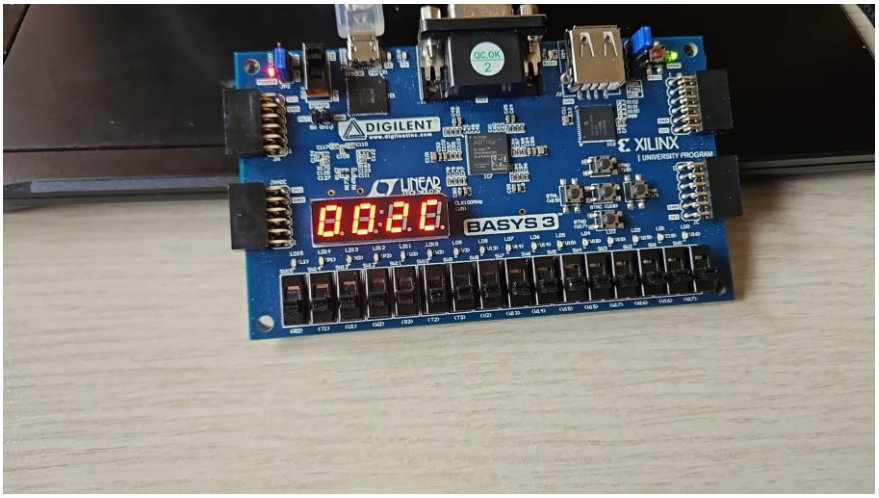


Figure 9: “002c” after 44 seconds

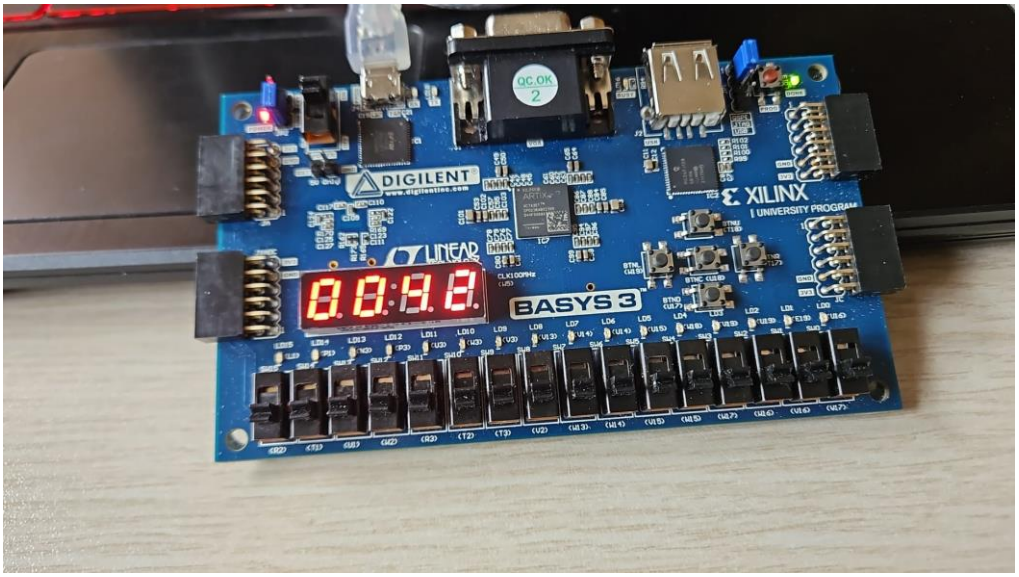


Figure 10: “0042” after 66 seconds

## Conclusion

The aim of this experiment was to create a seven-segment display using VHDL and create a visual effect called “persistance of vision”. I studied the connections of the seven-segment display. Then after finding a sample code for the clock dividing, I wrote the code for decoder and multiplexer to convert binary numbers into hexadecimal and distribute the signals for the proper Cathode of the LEDs. The results were as expected, and the display was successful. I learned a lot about the connections of the LED and counters. Also, it was the first time that I implemented the

decoder and multiplexer to convert the digits and use them to display the hexadecimal counting numbers on the LED.

## Code

### Main_module.vhd

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


entity main_7segment is Port ( clk: in std_logic; -- clock input reset: in std_logic; -- reset input cat: out std_logic_vector(3 downto 0); anode: out std_logic_vector(6 downto 0) ); end main_7segment;


architecture rtl of main_7segment is signal phase_counter: std_logic_vector(1 downto 0); signal sec_enable: std_logic; signal number: std_logic_vector(15 downto 0); begin


clock: entity work.segment_clock(rtl_clock) port map(clk => clk, reset => reset, phase_counter => phase_counter, second_cycle => sec_enable, sec_en => number);


driver: entity work.MUX(multiplexing) port map(clk => clk, reset => reset, phase_counter => phase_counter, number => number, sec_en => sec_enable, anode => anode, cathode => cat); end rtl;
```

### Segment_clock.vhd

```
-- Company: -- Engineer: -- -- Create Date: 03/26/2025 09:02:50 AM -- Design Name: -- Module Name: segment_clock - Behavioral -- Project Name: -- Target Devices: -- Tool Versions: -- Description: -- -- Dependencies: -- -- Revision: -- Revision 0.01 - File Created -- Additional Comments: --


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 UNISIMlibrary IEEE; use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_UNSIGNED.ALL; use IEEE.NUMERIC_STD;

entity segment_clock is Port ( clk: in std_logic; reset: in std_logic; clk_counter: out std_logic_vector(27 downto 0); phase_counter: out std_logic_vector(1 downto 0); sec_en: out std_logic_vector(15 downto 0); second_cycle: out std_logic); end segment_clock;

architecture rtl_clock of segment_clock is signal clk_counter1: std_logic_vector(27 downto 0); signal binary: std_logic_vector(15 downto 0):= (others => '0'); begin

process(CLK) begin if(reset = '1') then clk_counter1 <= (others => '0'); binary <= (others => '0'); else if rising_edge(CLK) then clk_counter1 <= clk_counter1 + '1'; if clk_counter1 =x"98968A" then binary <= binary + '1'; clk_counter1 <= (others => '0'); end if; end if; end if; end process;

second_cycle <= '1' when clk_counter1 =x"5F5E0FF" else '0'; phase_counter <= clk_counter1(1 downto 0); clk_counter <= clk_counter1; sec_en <= binary;

end rtl_clock;

**MUX.vhd**

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

entity MUX is Port (clk: in std_logic; reset: in std_logic; phase_counter: in std_logic_vector(1 downto 0); number: in std_logic_vector(15 downto 0); sec_en: in std_logic; cathode : out std_logic_vector(3 downto 0); anode: out std_logic_vector(6 downto 0) ); end MUX;

architecture multiplexing of MUX is signal comb: std_logic_vector(3 downto 0); begin

process(phase_counter) begin case phase_counter is when "00" => cathode <= "0111"; -- 1st digit comb <= number(15 downto 12); when "01" => cathode <= "1011"; -- 2nd digit comb <= number(11 downto 8); when "10" => cathode <= "1101"; -- 3rd digit comb <= number(7 downto 4); when "11" => cathode <= "1110"; -- 4th digit comb <= number(3 downto 0); when others => cathode <= "1111"; end case; end process;

LED: entity work.decoder(dec) port map(led_combination => comb, hexadecimal => anode);

end multiplexing;

**Decoder.vhd**

library IEEE; use IEEE.STD_LOGIC_1164.ALL;

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

architecture dec of decoder is begin process(led_combination) -- label the process begin case led_combination is when "0000" => hexadecimal <= "0000001"; when "0001" => hexadecimal <= "1001111"; when "0010" => hexadecimal <= "0010010"; when "0011" => hexadecimal <= "0000110"; when "0100" => hexadecimal <= "1001100"; when "0101" => hexadecimal <= "0100100"; when "0110" => hexadecimal <= "0100000"; when "0111" => hexadecimal <= "0001111"; when "1000" => hexadecimal <= "0000000"; when "1001" => hexadecimal <= "0000100"; when "1010" => hexadecimal <= "0000010"; when "1011" => hexadecimal <= "1100000"; when "1100" => hexadecimal <= "0110001"; when "1101" => hexadecimal <= "1000010"; when "1110" => hexadecimal <= "0110000"; when "1111"

=> hexadecimal <= "0111000"; when others => hexadecimal <= "1111111"; end case; end process; -- close process here

end dec;