Student Name: C3C Benedict J. Zito
Course: ECE 281 – Digital Design and Computer Architecture
Lab 2: Seven Segment Decoder
Instructor: Lt Col Jason M. Wyche
Section: M4
Date: 3 March 2026

Introduction

The objective of this lab was to use VHDL in Vivado to create a seven-segment decoder program that took a binary value input and produced a hexadecimal output. A decoder is a digital circuit that has N number of inputs and selects one of 2^N outputs. Each input and output can have multiple bits; in this lab, our decoder took an input of four bits (16 possible outputs) and produced an output of seven bits. The selected output was passed to a seven-segment display – a display of seven individual cathodes connected to a common anode. Each bit value of the output corresponded to the voltage of one of the cathodes in the seven-segment display. Because the anode in this implementation was "active-low" a bit value of '0' caused a high voltage for the cathodes.

The design was coded in VHDL and implemented onto the Basys 3 FPGA development board. The four-digit binary input was realized through four of the board's switches, decoded, and produced an output using the board's seven-segment LED display.

Methodology

VHDL Implementation

A behavioral VHDL model was developed to implement the seven-segment decoder design. The decoder assigned one of 16 seven-bit outputs to each possible 4-digit input using when-else statements. (Figure 1.)

An entity was declared with std_logic and std_logic_vector inputs and outputs. (Figure 2.) A seven-segment decoder component was declared with appropriate inputs and outputs. Wire signals were declared with std_logic and std_logic_vector to pass signals from the button to anode, and from the decoder output to the seven cathodes, respectively. (Figure 3.) Because the order of the cathodes in the basys3 board mirrored the original truth-table design with respect to most significant bit, the respective wires of the output were mapped 'backwards' to the respective cathodes of the seven segment display. (Figure 4.)

The behavioral model was chosen for ease of understanding and implementation of the decoder circuit.

Key Code Excerpts:

- Decoder assignment in sevenseg_decoder.vhd
- top_basys3 entity declaration
- Component and signal declaration in top_basys.vhd architecture

- Wire mapping to appropriate input, output ports; active-low and mirrored output values accounted for

```
with i_Hex select
    o_seg_n <=  "1111110" when "0000",
                "0110000" when "0001",
                "1101101" when "0010",
                "1111001" when "0011",
                "0110011" when "0100",
                "1011011" when "0101",
                "1011111" when "0110",
                "1110000" when "0111",
                "1111111" when "1000",
                "1110011" when "1001",
                "1110111" when "1010",
                "0011111" when "1011",
                "0001101" when "1100",
                "0111101" when "1101",
                "1001111" when "1110",
                "1000111" when "1111",
                "0000000" when others;
```

Figure 1. Decoder output assigned using when-else statements

```
entity top_basys3 is
    port(
        -- 7-segment display segments (cathodes CG ... CA)
        seg     :   out std_logic_vector(6 downto 0);  -- seg(6) = CG, seg(0) = CA

        -- 7-segment display active-low enables (anodes)
        an      :   out std_logic_vector(3 downto 0);

        -- Switches
        sw      :   in  std_logic_vector(3 downto 0);

        -- Buttons
        btnC    :   in  std_logic
```

Figure 2. Entity top_basys3 is declared with inputs switches and center button, outputs of anode and segments (cathodes)

```
component sevenseg_decoder is
    Port ( i_Hex : in STD_LOGIC_VECTOR (3 downto 0);
           o_seg_n : out STD_LOGIC_VECTOR (6 downto 0));
end component sevenseg_decoder;

  -- create wire to connect button to 7SD enable (active-low)
    signal w_7SD_EN_n   :   std_logic;
    signal w_seg_n      :   std_logic_vector(6 downto 0);
```

Figure 3. Component sevenseg_decoder is declared with binary input and seven bit output

```
w_7SD_EN_n  <= not btnC;

seg(6) <= not w_seg_n(0);
seg(5) <= not w_seg_n(1);
seg(4) <= not w_seg_n(2);
seg(3) <= not w_seg_n(3);
seg(2) <= not w_seg_n(4);
seg(1) <= not w_seg_n(5);
seg(0) <= not w_seg_n(6);

an(0)   <= w_7SD_EN_n;
an(1)   <= '1';
an(2)   <= '1';
an(3)   <= '1';
```

Figure 4. Wires used to assign reciprocal values

from decoder output to active low input.

Anode 0 is active when center button pressed (active-low)

Anodes 1, 2, 3 are bypassed.