Lab 3
Binary and Seven-Segment Decoders

Name_____ Class _____ Date _____

OBJECTIVES:

Upon completion of this laboratory exercise, you should be able to:

- Enter the design for a binary decoder in Quartus II as a Block Diagram File or a VHDL design entity.
- Create a Quartus II simulation of a binary decoder.
- Use VHDL to create a seven-segment decoder.
- Test the binary and seven-segment decoders on the Altera DE1 or DE2 board or equivalent.

REFERENCE READING:

Dueck, Robert K., *Digital Design with CPLD Applications and VHDL, 2/e*:
Chapter 6: Combinational Logic Functions
6.1 Decoders

EQUIPMENT REQUIRED:

Altera DE1 or DE2 Development and Education Board
AC Adapter, minimum output: 7 VDC, 250 mA DC
USB Blaster Cable
Quartus II Software
Anti-static wrist strap

EXPERIMENTAL NOTES:

VHDL Implementation of a Binary Decoder

The most straightforward way to implement a binary decoder in VHDL is to use a selected signal assignment statement. In this construct, the four outputs of a 2-line-to-4-line decoder are defined as a vector for each combination of inputs, also expressed as a vector.

For example, for inputs $D_1D_0 = 01$, output Y_1 will be HIGH and the remaining outputs are LOW. The output vector is written as 0010, where the leftmost bit is Y_3 and the rightmost bit is Y_0, as defined in the entity declaration. (The output vector could also be defined with Y_0 on the left, by changing the port definition to read

```
y : OUT STD_LOGIC_VECTOR (0 to 3).
```

In this case, the output vector for input 01 is 0100, since Y_1 is now second from the left.)

The VHDL design entity shown below illustrates the use of a selected signal assignment to define a decoder. The **others** clause is required since the STD_LOGIC type encompasses values other than 0 and 1. The default value in the **others** clause shows the outputs all inactive (0000) for any unspecified input value.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY dcd2to4 IS
   PORT(
      d   : IN    STD_LOGIC_VECTOR (1 downto 0);
      y   : OUT   STD_LOGIC_VECTOR (3 downto 0));
END dcd2to4;

ARCHITECTURE decoder OF dcd2to4 IS
BEGIN
   WITH d SELECT
      y <=    "0001" WHEN "00",
              "0010" WHEN "01",
              "0100" WHEN "10",
              "1000" WHEN "11",
              "0000" WHEN others;
END decoder;
```

Seven-Segment Decoder

A seven-segment display, shown in Figure 1, consists of seven luminous segments, such as LEDs, arranged in a figure-8 pattern. The segments are conventionally designated *a* through *g*, beginning at the top and moving clockwise around the display.

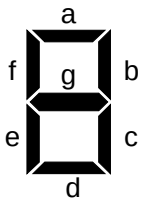

Figure 1 Seven-segment Display

When used to display decimal digits, the various segments are illuminated as shown in

Figure 2. For example, to display digit 0, all segments are on, except *g*. To display digit 1, only segments *b* and *c* are illuminated.

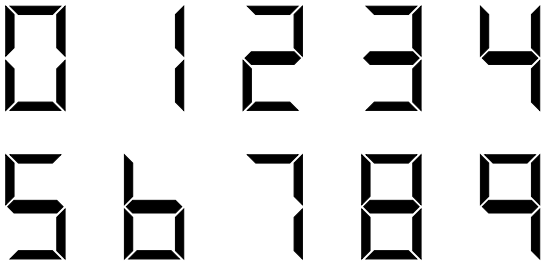

Figure 2 Digit patterns for BCD-to-seven-segment display

The seven-segment displays on the Altera UP-2 and DE2 boards are configured as common-anode, meaning that the anodes of all LEDs are tied together and connected to the board power supply, V_{CC}. (Refer to Figure 6.17, p. 282 in *Digital Design with CPLD*

Applications and VHDL.) To turn on a segment, the cathode end of the LED is set to logic 0 through a current-limiting series resistor. This is illustrated for digits 0 and 1 in the partial truth table shown in Table 1.

				a	***b***	***c***	***d***	***e***	***f***	***g***
D_3	***D_2***	***D_1***	***D_0***	***Y_0***	***Y_1***	***Y_2***	***Y_3***	***Y_4***	***Y_5***	***Y_6***
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1

Table 1 Partial Truth Table for a Common Anode BCD-to-7-Segment Decoder

The seven-segment displays and series resistors on the Altera DE1 and DE2 boards are hardwired to the board's FPGA. Thus, all that is required to turn on the illuminated segments for each digit is to make the appropriate FPGA pins. A VHDL file for a common anode BCD-to-seven-segment decoder is shown below.

```
-- bcd_7seg.vhd
-- Common Anode BCD-to-seven-segment decoder
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY bcd7seg IS
   PORT(
       d   : IN    STD_LOGIC_VECTOR(3 downto 0);
       y   : OUT   STD_LOGIC_VECTOR(0 to 6));
END bcd7seg;

ARCHITECTURE seven_segment OF bcd7seg IS
BEGIN
   WITH d SELECT
       y <=    "0000001" WHEN "0000", -- display 0
               "1001111" WHEN "0001", -- display 1
                   -- several lines missing here
               "0001100" WHEN "1001", -- display 9
               "1111111" WHEN others;

END seven_segment;
```

Lab 3
Binary and Seven-Segment Decoders

PROCEDURE:

VHDL Implementation of a Binary Decoder

1. Write a VHDL file for a 3-line-to-8-line decoder with active-HIGH outputs. Use a selected signal assignment statement. Save the file as **g:\qdesigns\dgs\lab03\dcd3to8.vhd** and use the file to create a new project. Compile the design.

2. Write a test spec for the 3-line-to-8-line decoder you created in step 1 of this procedure.

Test Spec:

Instructor's Initials_____

3. Use the test spec to create a simulation in ModelSim to test the correctness of your design. Show the simulation to your instructor.

Instructor's Initials_____

4. Assign pin numbers to the VHDL decoder, as shown in Table 3. Compile the design again and download it to the FPGA test board. Demonstrate the decoder operation to your instructor. ***After demonstrating the decoder, close the Quartus II project.***

Instructor's initials_____

Board: DE1 or DE2		
Family: Cyclone II		
Device: EP2C20F484C7 (DE1) or EP2C35F672C6 (DE2)		
Pin Name	**Pin Number**	**Switch or LED Number**
d[0]		
d[1]		
d[2]		
y[0]		
y[1]		
y[2]		
y[3]		
y[4]		
y[5]		
y[6]		
y[7]		

Lab 3
Binary and Seven-Segment Decoders

Table 3 Pin Assignments for 3-to-8-line Decoder

Hexadecimal-to-Seven-Segment Decoder

1. Create a new folder called **g:\qdesigns\dgs\lab03\hex7seg\.** Create a VHDL file in this folder that describes a hexadecimal-to-seven-segment decoder, using the segment patterns in Figure 3 as a model. Note that the patterns for digits 6 and 9 are shown differently than in Figure 2. Save the VHDL file in your new folder as **hex7seg.vhd** and use the file to create a new project.

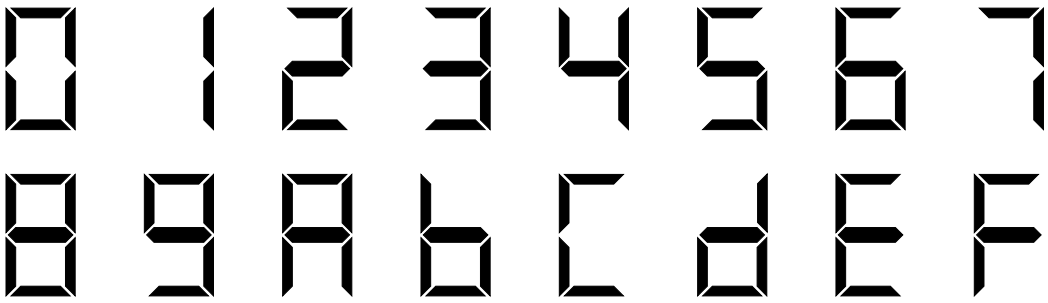

Figure 3 Hexadecimal Digit Pattern for a Seven-Segment Display

2. Assign pin numbers in Table 4, using the DE1 or DE2 User Manual on the course sharepoint. Compile the file and download it to the DE1/DE2 board. Demonstrate its operation to your instructor.

Board: DE1 or DE2		
Family: Cyclone II		
Device: EP2C20F484C7 (DE1) or EP2C35F672C6 (DE2)		
Pin Name	**Pin Number**	**Switch or Display Segment number**
d[0]		
d[1]		
d[2]		
d[3]		
a		
b		
c		
d		
e		
f		
g		

Table 4 Pin Assignments for Hex-to-Seven-Segment Decoder

Instructor's Initials_____

Lab 3
Binary and Seven-Segment Decoders

Bonus Problem

You are eligible to receive a 25% bonus for completing the following problem by the stated lab deadline. To receive this credit, the remainder of the lab must be completed by the stated deadline.

Create a 4-digit display that will suppress leading zeroes, but display internal zeroes, as shown in Figure 4. Figure 5 (next page) shows the connections between the display decoders, as implemented in a Quartus II Block Diagram File. Refer to pages 287-291 in the textbook.

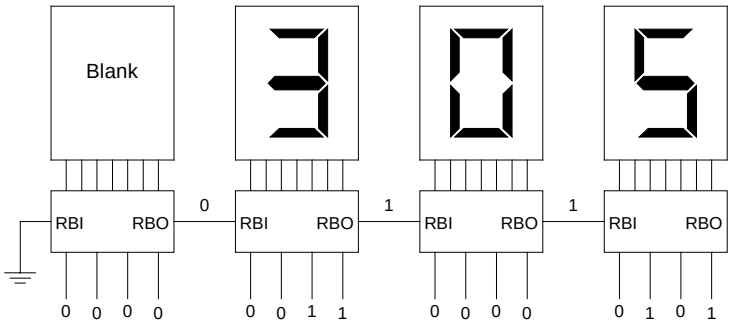

Figure 4 4-Digit Display with Leading-Zero Suppression

To receive credit for this problem:

- Write the VHDL file for the decoder.
- Create a Block Diagram File, similar to the one in Figure 5, where the decoder symbol is created from your VHDL file.
- Look up pin numbers and assign them to the design.
- Demonstrate correct operation of the design to your instructor.

Instructor's Initials _____

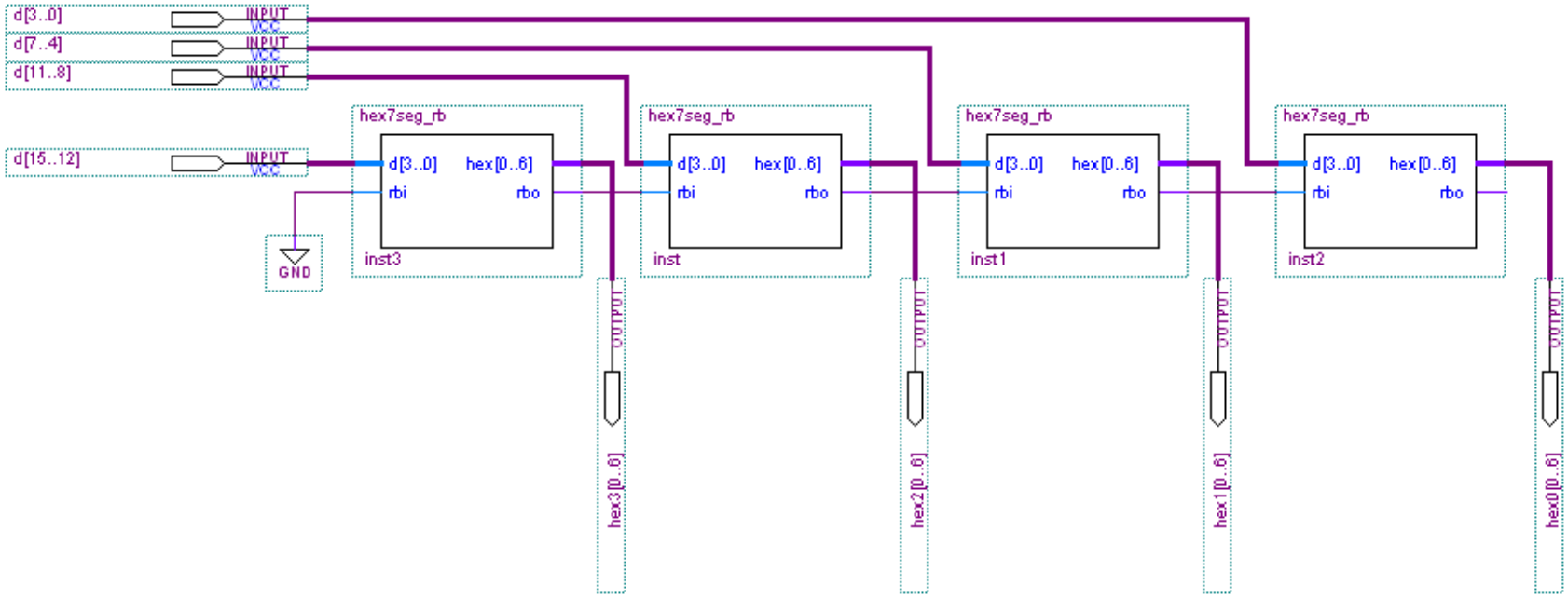

Figure 5 Quartus II Block Diagram for 4-Digit Display with Leading-Zero Suppression