**Lab 4: Designing a 7-Segment Display Decoder and ALU** **Name:** Luke Bachman

EGCP 281: Designing with VHDL **Date:** Thursday, November 7, 2019

***You may discuss this assignment with others, but this assignment must be completed individually.***

*Please recall that academic dishonesty will not be tolerated. By submitting this assignment, you understand penalties will be assessed if you submit work for credit that is not your own.*

**Due Date: Upload this work sheet to Titanium by Thursday, October 24, 2019 by 1:00 PM**

**Grade** **Grade Item**

/3 Task 1: Design of a Seven-Segment Display Decoder Circuit

/3 Task 2: Design and Simulation of a behavioral 4-bit Arithmetic Logic Unit (ALU)

Task 3: Combining your ALU and Seven-Segment Display Decoder Circuit

*/3 Demonstration of Task 3*

/6 Lab Recap Questions

**/12 Total Score (of deliverables and questions)**

**/3 Lab Demonstrations**

**Once you have completed the lab (or once you are done for the day), and after you close your project, remember to copy your project folder to a flash drive, your Dropbox or Google Drive, or your email in order to keep a copy of your files.**

**Recommended Reading:** *Digilent Real Digital Module 6 and 7*

**Objectives:**

* Learn and apply more advanced VHDL statements
* Gain more experience with modular circuit design
* Gain experience working with a seven-segment display
* Use behavioral arithmetic operators in a VHDL design
* Gain experience with Arithmetic Logic Units (ALUs)

**Lab Description:**

You will design a seven-segment display decoder to translate a hexadecimal number into the signals used to control the seven-segment display (the digit display circuitry) on the FPGA board. This is a very common output device that we will use repeatedly in class. You will then create a behavioral design of an Arithmetic Logic Unit (ALU). This is a very common circuit used in processors and other digital systems to make arithmetic and logical calculations. You will then combine your seven-segment display decoder with your ALU; you will do this by using modular design.

**Lab Tasks:**

1. **Create a seven-segment display decoder circuit**This is a very useful circuit and we will use this in a future lab (so make sure you save a copy of your design module)
   1. This seven-segment display must display **hexadecimal** digits. Illuminate the seven-segment display with the numerals 0-9 as well as the letters **A, b, C, d, E,** and **F**. Remember, each LED segment is **active low** (requires a logic 0 to illuminate).
   2. **Your seven-segment display decoder circuit will use two busses.** Use a 4-bit bus (controlled by four switches) as your input and use a 7-bit bus (which will control the seven-segment display) as your output.
   3. Create a new project and implement this circuit using VHDL. Rather than writing seven assignment statements based on the Boolean expressions of the segments, use an **advanced VHDL statement** (such as a when or with-select statement) and **follow your truth table from the prelab problems** to design your module. Make corrections to your truth table as necessary.
   4. [3 pts] While you certainly could create a VHDL test bench and all possible input signal combinations, it is fairly easy to just test this on the FPGA board. Implement your circuit on the FPGA board. Connect the input bus to four switches and connect the output bus to the seven cathodes for the numeric segments—make sure you connect the correct bits of the output bus to the correct A-G segments (if you connect these backwards, then your numbers will not show correctly—they may appear somewhat “mirrored”). Verify that your circuit operates correctly on the FPGA board. If your output behavior is correct, then copy your VHDL design code and answer the questions below:
      1. Place your VHDL design module code here (copy-paste as text, not an image):

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/30/2019 11:20:06 AM

-- Design Name:

-- Module Name: Decoder - 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 UNISIM;

--use UNISIM.VComponents.all;

entity decoder is

Port ( Input : in STD\_LOGIC\_VECTOR (3 downto 0);

Output : out STD\_LOGIC\_VECTOR (6 downto 0));

end decoder;

architecture Behavioral of decoder is

begin

WITH Input SELECT

Output <=

"0000001" WHEN "0000", --0

"1001111" WHEN "0001", --1

"0010010" WHEN "0010", --2

"0000110" WHEN "0011", --3

"1001100" WHEN "0100", --4

"0100100" WHEN "0101", --5

"0100000" WHEN "0110", --6

"0001111" WHEN "0111", --7

"0000000" WHEN "1000", --8

"0000100" WHEN "1001", --9

"0001000" WHEN "1010", --A

"1100000" WHEN "1011", --b

"0110001" WHEN "1100", --C

"1000010" WHEN "1101", --d

"0110000" WHEN "1110", --E

"0111000" WHEN "1111", --F

"1111111" WHEN OTHERS; --blank when not a digit

end Behavioral;

* + 1. Is this circuit working correctly on the FPGA board? Please explain why or why not:

The code does work correctly on the FPGA board. It lights up all of the correct parts of the display for every input combination of the four switches.

* + 1. Did you have any difficulties obtaining the correct output on the FPGA board (e.g. did you have to debug your circuit or constraints file)? Did you have to make any changes to your truth table from the prelab assignment? Please explain your answers.

I did not have any difficulties obtaining the correct output on the FPGA board. Made my truth table correct after the pre-lab and was able to implement what I had much easier.

1. **Create a behavioral VHDL module for a 4-bit Arithmetic Logic Unit (ALU)**
   1. This 4-bit ALU will calculate arithmetic and logical expressions on two 4-bit numbers. Use behavioral expressions for the arithmetic and logic expressions (do not use port map statements to create a structural design using your previous ripple-carry adder). Design the ALU using the select input/operation code (opcode) definitions from the following table:

|  |  |
| --- | --- |
| **Opcode** | **Function** |
| 000 | A |
| 001 | A and B |
| 010 | A or B |
| 011 | A xor B |
| 100 | 0 |
| 101 | not A |
| 110 | A plus B |
| 111 | A mult. by 2 |

* 1. Use your current project and create a new VHDL design module to implement this ALU. Do not create a structural design by using bit slices of an ALU. Instead, create a behavioral ALU design (see the example in *Digilent Real Digital Module 7*). Use an **advanced VHDL statement** (such as a when or with-select statement) to design your ALU.  
       
     Note, when you use the behavioral operators for addition (+) or subtraction (-), you will need to include an additional library at the top of your design module. For the ALU, you should be using the following libraries:  
       
     **library IEEE;  
     use IEEE.STD\_LOGIC\_1164.ALL;  
     use IEEE.STD\_LOGIC\_UNSIGNED.ALL; -- add this library for the ALU**
  2. [3 pts] Create a VHDL test bench to test your ALU. Use two input signal (the 4-bit values for A and B) combinations to test each operation of the ALU (you’ll have 16 total test cases). Simulate your design and verify your output. If your output behavior is correct, then copy your VHDL design code and simulation screenshot below:
     1. Place your VHDL design module code here (copy-paste as text, not an image):

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/30/2019 12:48:49 PM

-- Design Name:

-- Module Name: ALU - 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;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL; -- add this library for the ALU

-- 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 UNISIM;

--use UNISIM.VComponents.all;

entity ALU is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

OP : in STD\_LOGIC\_VECTOR (2 downto 0);

Output : out STD\_LOGIC\_VECTOR (3 downto 0));

end ALU;

architecture Behavioral of ALU is

begin

with OP select

Output <= A when "000",

(A and B) when "001",

(A or B) when "010",

(A xor B) when "011",

"0000" when "100",

not(A) when "101",

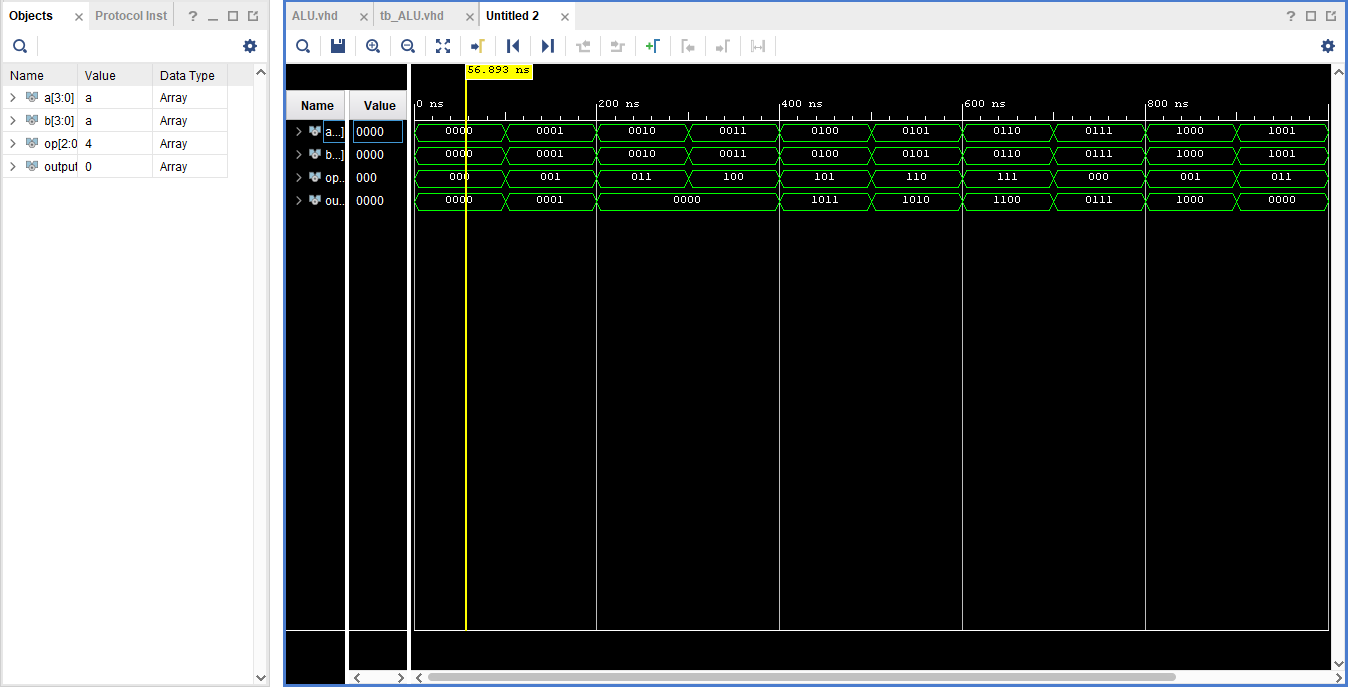
(A + B) when "110",

A(2 downto 0) & '0' when "111", --(A \* 2)

(others => '0') when others;

end Behavioral;

* + 1. Place your screenshot here (enlarge your waveform and do not crop the image (it should show the whole screen) or you will lose points):



* + 1. Explain why your circuit is (or is not) working correctly:

My circuit appears to be working for all the operations that I implemented on the ALU. Using the opcode, the correct instruction is performed based off our instruction table. Using the test bench I made, I tested 2 four bit inputs for A and B in the ALU and the operations that were performed resulted in the correct output during the simulation.

1. **Combine your ALU with your seven-segment display decoder**Lastly, it would be really convenient to implement your ALU on the FPGA board and use the seven-segment display decoder you created from task 1. Luckily, this is possible. Follow these steps:
   1. Use your current project and create a new VHDL design module to implement this circuit using VHDL. In this module, you will use port map statements to include your designs from tasks 1 and 2. Use VHDL signals to connect these two components together.
   2. [3 pts] Again, while you certainly could create a VHDL test bench and all possible input signal combinations, it is fairly easy to just test this on the FPGA board. You have already tested each component (your ALU and seven-segment display decoder) individually; the whole system should work ok as long as you correctly connect them together. Implement your circuit on the FPGA board. Connect all of the inputs of your ALU to switches (make sure you know which switches represent A, B, and the opcode) and connect the output bus to the seven cathodes for the numeric segments. Verify that your circuit operates correctly on the FPGA board.
      1. Place your VHDL design module code (of the entire ALU/seven-segment display decoder circuit) here (copy-paste as text, not an image):

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 11/07/2019 09:09:25 PM

-- Design Name:

-- Module Name: ALU\_decoder - 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 UNISIM;

--use UNISIM.VComponents.all;

entity ALU\_decoder is

Port (Input : in STD\_LOGIC\_VECTOR(3 downto 0);

Output: out STD\_LOGIC\_VECTOR(3 downto 0);

OP : in STD\_LOGIC\_VECTOR (2 downto 0);

A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0) );

end ALU\_decoder;

--This is the top file shell where we define ports only

--These are also known as your global inputs or main inputs to the overall circuit

--------------------------------------------------------------

architecture Behavioral of ALU\_decoder is

component ALU

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

OP : in STD\_LOGIC\_VECTOR (2 downto 0);

Output : out STD\_LOGIC\_VECTOR(3 downto 0));

end component;

component decoder

Port( Input : in std\_logic\_vector(3 downto 0);

Output : out std\_logic\_vector(6 downto 0));

end component;

--This is where we delare the components we are using

----------------------------------------------------------------

signal int\_signal : STD\_LOGIC\_VECTOR;

--This is the signal that connects component a to component b

--The 4-bit ALU output is going to the input of the decoder

begin

component\_a: ALU port map(

A => A,

B => B,

OP => OP,

Output => int\_signal );

component\_b: decoder port map(

Input => int\_signal,

Output => Output );

end Behavioral;

* + 1. Ask the instructor to check the FPGA board implementation of your circuit

Now complete the lab recap questions below and submit the worksheet on Titanium by the due date.

**Lab Recap Questions:**

Type your answers to these questions once you finish the lab (do not print and write these):

1. [2 pts] Imagine that you only used assignment statements for the design of the seven-segment display decoder. How would you obtain the Boolean expressions for the seven segments? What would your VHDL design module code look like? Which way do you prefer designing the seven-segment display decoder, this way or by using the advanced VHDL statements you used in task 1?

If I were to just use assignment statements and obtain Boolean expressions for the seven segments, I would have to make 7-Kmaps and come up with the expressions. The VHDL code would be an absolute mess if there were all those different Boolean expressions, and it would be much harder to debug too. I much prefer the advanced VHDL statements because the code has a lot more readability and cuts out the work of coming up with Boolean expressions, I just have to make a truth table.

1. [2 pts] Why might using advanced VHDL statements, such as the “when” or “with-select” statements be preferable to using assignment operations (e.g. “F <= (A or B) and C;”) in VHDL? In addition, give an example of a situation where using an assignment operation in VHDL would be preferable over advanced statements.

Using these advanced VHDL statements are preferable to using assignment operations when you start getting way larger circuits with lots of inputs and outputs. The assignment operation would be preferable over advanced statements when dealing with small basic circuits with only a few inputs and outputs. An example would be when you are trying to implement a basic logic gate. There is no need for an advanced VHDL statement in that situation.

1. [2 pts] Why are ALUs important—what do these do? What types of circuits/systems use ALUs?

ALUs are super important because they are what perform all of the logical and arithmetic operations in a system! ALUs are used in many things like calculators and most importantly, CPUS. These circuits are absolutely essential for making calculations.

1. List the references you used for this lab assignment (e.g. sources/websites used or students with whom you discussed this assignment)

I used Digilent Real Digital, Lecture Slides, Allaboutfpga.com, VHDL portmapping example

1. Do you have any comments or suggestions for this lab exercise?

As weird as this sounds, building the first two components were the east part for me. The two things that I struggled with the most in this lab were creating the Multiplication operation in the ALU and portmapping the components together. It took a lot of time to figure these two things out. I think that you should say in the instructions for the ALU that you need to shift your vector to the left one bit by using bit slicing and concatenation for the multiplication instruction. The other thing was wiring the ALU to the decoder. I struggled putting this into code. I knew how to draw it, but it took awhile before I found a good example on how to use VHDL portmapping with two components. I wish you included an example for this.