Cpr E 281 LAB3
ELECTRICAL AND COMPUTER
ENGINEERING
IOWA STATE UNIVERSITY

Introduction to Circuit Design using
Verilog

PRELAB!
Read the entire lab, and **complete** the prelab questions (Q1-Q2) on the answer sheet **before** coming to the laboratory.

1.0 Objectives

In the last lab we learned how to use schematic capture to design digital circuits in Altera Quartus Prime. The steps performed were: understanding the problem statement, deriving a logic function, drawing the logic expression using logic gates (NOT, OR, AND), naming the inputs and the output, and finally using an FPGA to verify the circuit for all possible values of the input. This lab is similar. However, we will take a different approach. We will write **Verilog** code to describe the logic expression instead of drawing the logic gates.

2.0 Setup

2.1 Verilog

At this point you might be asking yourself what is Verilog anyway? First you have to know what a Hardware Description Language (HDL) is. HDL's are used to perform a logic synthesis of a design. That means we write a code describing our logic and the HDL compiles it to create the adequate circuit diagram. This simplistic definition is very powerful and tells us that now we can just "code" our logic, and we will get the equivalent circuit, without going through the hassle of drawing it. Verilog is a HDL that has started being widely used in the US industry since it went open-source in 1990. It is portable in nature, meaning that the same code can be used in different CAD tools. Just like any coding language it has a compiler, libraries, syntax, commenting etc. There exist many HDL's besides Verilog. The most popular among others is VHDL (V stands for Very High Speed Integrated Circuit), and is used in most industries in the rest of the world.

More information on Verilog can be found in http://www.verilog.com/, and in your textbook.

2.2 Structural coding

When we look at the structural level of a design, we are interested in how each variable is processed before a design produces an output. For example, in implementing the expression $F = A + B \cdot \overline{C}$, we will first **INVERT** C, **AND** it with B, and then **OR** the result with A.

Cpr E 281 LAB3
ELECTRICAL AND COMPUTER
ENGINEERING
IOWA STATE UNIVERSITY

Introduction to Circuit Design using
Verilog

The Verilog code for F is as follows:

```
module structural_example (F, A, B, C);
        input A, B, C;
        output F;              /*always comment your code
                               so you know what you did!*/
                               /* This code define the structure of the circuit */
                               /* First symbol is always output for not, and, and or */
        not (D,C);             //C is inverted and stored in D, output is named
        and (E,B,D);           //B AND with D, store result in E, output is named
        or (F,A,E);            //A OR with E, store result in F

endmodule
```

2.3 Behavioral Coding

Another way to generate a circuit is to study its behavioral description instead of its structural description. This implies that we are looking at our logic design from a system point of view, i.e. what the inputs are, what the outputs are, and how they are related to each other. This is usually a lot simpler to do than the structural description. It is faster in user time to design. Considering our previous example again, we could write the following code:

```
module behavioral_example (F, A, B, C);
        input A, B, C;
        output F;      /*always comment your code
                       so you know what you did!*/

        assign F=(A|(~C&B));  //write down the logic expression

endmodule
```

After this brief introduction to Verilog, we will get started on two design examples. For a more thorough reference on Verilog, read textbook.

2.4 Setup

Create a folder **U:\Documents\CPRE281\Lab03**, and four sub-folders **\Lab03\lab3step0**, **\Lab03\lab3step1**, **\Lab03\lab3step2**, and **\Lab03\lab3step3**. You will be saving your work and programming the hardware for this lab in these directories.

3.0 Design 1

You will design and verify the "farmer's problem" describe below using Verilog. You will use the product-of-sums (POS) expression. You will run a structural and a behavioral description of the same problem. The goals of this lab session are:

- o To learn use of Schematic Capture using POS expression.
- o To learn use of Verilog Structural description.
- o To learn use of Verilog Behavioral description.

For each design simulation, you have to show your simulation results and your code and/or schematic to the lab instructor to obtain his/her initials on your answer sheet.

Description:

A farmer owns two barns; one north of a creek and the other south. The farmer has a Cabbage, a Goat, and a Wolf. The Farmer needs to put each item in a barn every night. If the Cabbage and the Goat are in the same barn, the Goat will eat the Cabbage. If the Wolf and the Goat are in the same barn, the Wolf will eat the Goat. The Farmer is worried and you have to design an alarm circuit that will let him know if two items can safely be placed in a barn.

For this circuit, you have three $inputs = \{Cabbage, Goat, Wolf\}$ and one $output = \{Alarm\}$.

If an input is in the north barn, it gets assigned logic 1, and if it is in the south barn it gets assigned logic 0. The output Alarm, asserts if there are two items in a barn that should not be kept together.

---

1. After verifying your logic expression, use schematic capture to make a digital circuit similar to what you did in Lab02. Create a new project named *lab3step0.qpf*, and save your file under **\CPRE281\Lab03\lab3step0**. Verify that the results of your POS expression are correct using the DE2-115 board.

2. To write Verilog code, use a text editor that generates plain ASCII text. Quartus Prime has one built-in editor. Create a new project named *lab3step1.qpf*, and save your file under **\CPRE281\Lab03\lab3step1**.

3. Then add a text file by going to **File -> New** under **Design File** select **Verilog HDL File** and save it as **lab3step1.v** (*IMPORTANT: USE THIS NAME FOR YOUR FILE*).

4. Write a structural description for the farmer's problem similar to what was explained in section 2.2. Again you will have to use the POS. Also, **the name of your module function should be the same as the file name**! Otherwise, you will get an error.

5. When you are done writing your code, start the compilation. If you get errors, fix your code, and if needed, take a look at your textbook for more details on the syntax or ask your lab instructor.
6. Next assign the I/O pins in the same way as you did for the schematic version of your circuit. Choose a group of toggle switches for the inputs, and an LED for your output. A document is provided with the Lab03 files that contain tables that show which FPGA pins are connected to some of the DE2-115 hardware I/O devices (***DE2-115_PIN_ASSIGNMENTS.pdf***).
7. Compile one more time, then open the programmer tool, **Tools -> Programmer**, and program the DE2-115 with your new circuit.
8. Finally, verify that you are getting the exact same output as in the schematic capture. Show your code and results to the lab instructor.
9. Repeat the previous Verilog steps, except that this time you will use a <u>behavioral description</u>. Create a new project and name the Verilog file **lab3step2.v**. Look at section 2.3 for more info on the behavioral description.
10. Again, show your code and results to the lab instructor.

4.0 Design 2 (Lab3step3)

This time we will make the farmer's problem more complex by adding the farmer himself to the bunch. That way his presence will keep the wolf from eating the goat and the goat from eating the cabbage.

*   Write a canonical SOP expression, and then simplify it.
*   Use Verilog to create a digital circuit for the alarm. You can choose either the structural or the behavioral description (*HINT: use the one that uses fewer lines of code*).
*   Use the DE2-115 board to verify your circuit operates properly. Show your results to the lab instructor.

5.0 Complete

You are done with this lab. Ensure that all lab files are closed, exit Quartus Prime, log off the computer, power down the DE2-115 board, and hand in your answer sheet. **Don't forget to write down your name, student ID, and your lab section number**.