**ECEN 449 - Lab Report**

**Lab Number:** 3

**Lab Title:**

Creating a Custom Hardware IP and Interfacing it with Software

**Section Number:** 508

**Student's Name:** Samuel Fafel

**Date Due:** 03-02-2023

**TA:** Prajwal Holla

#### Purpose/Introduction:

This lab familiarized me with the creating and importation of custom IP modules, and integrating them into larger systems.

#### Procedure:

1. Created a new project and new block design called "multiply".
2. Created and packaged a new IP and imported it into the block design.
   1. Inside the IP, edited the generated Verilog code to remove all writing to slv\_reg2.
   2. Added new user logic to write to slv\_reg2.
3. Exported the completed block design to SDK and created a new C project in SDK.
4. Edited the provided "helloworld.c" file to perform multiplication of numbers using the block design.
5. Programmed the FPGA board and ran using picocom in a terminal window.
   1. Picocom showed the output for each requested multiplication.

#### Results:

The following is the last few lines of output from the Picocom terminal:

siv\_reg0 = 7

siv\_reg1 = 10

ans = 10 \* 7 = 70

siv\_reg0 = 8

siv\_reg1 = 10

ans = 10 \* 8 = 80

siv\_reg0 = 9

siv\_reg1 = 10

ans = 10 \* 9 = 90

siv\_reg0 = 10

siv\_reg1 = 10

ans = 10 \* 10 = 100

For the entire output, see [this document](https://docs.google.com/document/d/1AzaoWdjTbVFaaFbd70svyg1muKRNj9tKPjHt-PNGoCo/edit?usp=sharing).

#### Conclusion:

This lab was relatively simple, providing an overview of how to create and import a custom IP to a block design, and then use that block design to accomplish a task, such as multiplication.

#### Questions:

1. The tmp\_reg stores the current value of the multiplication of reg0 and reg1, so that it can then be stored in reg2. If tmp\_reg was removed, the values of reg0 and reg1 may change before the result of their multiplication is stored in reg2.
2. Values which multiply to a number greater than 32 bits would result in an overflow error. This can be corrected by either increasing the number of bits the result can hold (difficult) or creating a method by which to keep track of the overflow, and account for it after the operation is complete (easier).

#### Appendix:



