# CSEE W4840 Embedded System Design Lab 5

Stephen A. Edwards

Due March 2, 2006

**Abstract**

Implement a complex multiplier as an OPB peripheral. Write a small C program that exercises it (i.e., performs a number of operations and displays the results).

## 1 Introduction

One of the amazing things about using an FPGA is the ability to add any sort of hardware acceleration to your software programs you can imagine. In this lab, you'll implement a fairly simple example of this: a peripheral that can (quickly) multiply two complex numbers and return the results.

Complex multiplication is particularly useful in a number of signal processing algorithms, including the Fast Fourier Transformation. Specifically, the operation your peripheral is to perform is

$$(a+bi)(c+di) = (ac-bd)+(ad+bc)i = e+fi$$

where $a, b, \ldots, f$ are each eight-bit signed integers.

Notice that a single complex multiplication requires four normal multiplications, an addition, and a subtraction. Multipliers are large circuits, so a main challenge in this lab is to re-use the single multiplier we have given you to perform the four operations in different cycles.

The multiplier is a large, slow block, so it probably impossible to get through a multiplexer, the multiplier, an inverter, another mux, and an adder in a single cycle, so I suggest adding an intermediate register that holds the product so the addition can take place in the next cycle.

Below is a suggested block diagram.

It should be possible to do the complete multiplication in four cycles, following this schedule:

| cycle | multiplier | adder | registers |
|---|---|---|---|
| 1 | $ac$ | - | - |
| 2 | $bd$ | $ac+0$ | $\rightarrow e$ |
| 3 | $ad$ | $e+(-bd)$ | $\rightarrow e$ |
| 4 | $bc$ | $ad+0$ | $\rightarrow f$ |
| 5 | - | $f+bc$ | $\rightarrow f$ |

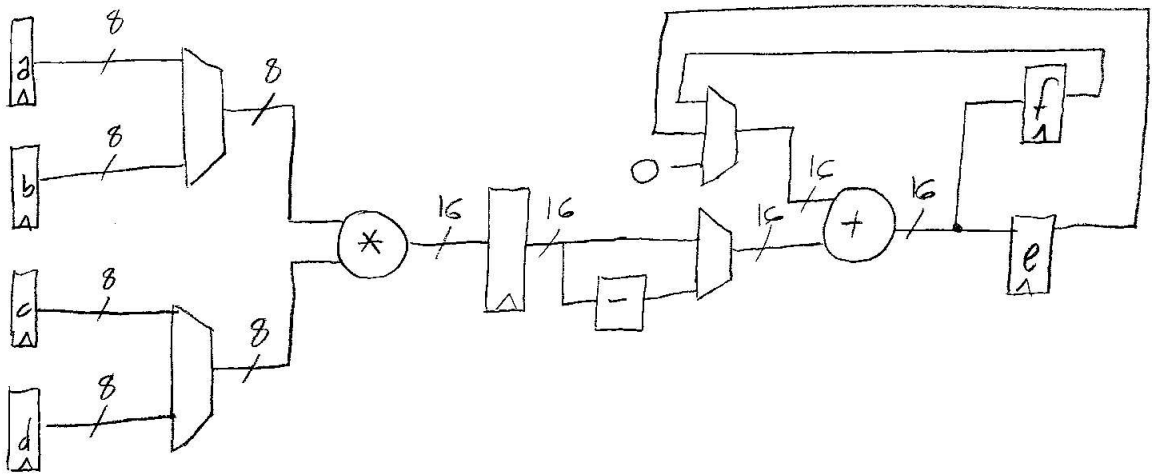
## 2 The Assignment

In lab5.tar.gz, you will find a template that includes an OPB peripheral called "opb_xsbcomplex_multiplier_v1_00_a." It instantiates an $8 \times 8 = 16$ multiplier that we created using a Xilinx tool called Coregen and illustrates how to write a simple peripheral that has registers that can be read and written.

At the moment, the multiplier is directly connected to two of the input registers and its output is connected so that it can be read by a C program.

Disconnect the multiplier, build the datapath, add an FSM that is triggered when a write operation occurs on any of the registers, and write a C program that demonstrates that your peripheral actually performs complex multiplication.

To design the FSM, list all the control signals you need (e.g., selection signals for the two muxes driving the multiplier, latch enable signals for the $e$ and $f$ registers) and devise a simple state machine that sets these signals as necessary to make the datapath execute this sequence of operations.

As usual, turn in a printout of *every* file you personally create for this assignment. Make sure your names are on it. Demonstrate it to a TA, have him sign off on it, and turn in the listing.

VHDL is a terribly verbose language that can quickly become unreadable. Again, we are looking for style and substance.