ComS / CprE 583 – Reconfigurable Computing
Homework #2
Assigned: September 6
Due: September 20 (12:00pm)

[Note from Joe: The four questions here relate directly to topics covered in the first three weeks of class. I have tried to make the expectations for each question clearer but much like in Homework #1, I am not going to be providing a direct path to any specific solution. Access to two commercial FPGA design tools (Xilinx ISE and Altera Quartus II) are needed to complete question 4. If you do not have access to the digital design lab in Coover Hall, then you must download and install both the Xilinx ISE Webpack (http://www.xilinx.com/webpack) and the Altera Quartus II Web Edition software (http://www.altera.com). Both of these tools require you to create a free web account and will take some time to download and install.]

1) We covered in some detail most of the various features used in the Xilinx series of FPGAs from the XCV3000 series all the way up to the current-day Virtex-4 and Virtex-5 families. On the flip side, we have only barely touched upon the Altera series of FPGAs, except to note that the base technology is very similar to that of Xilinx, with some changes in terminology.

(a) Provide a 1:1 mapping of the terminology used by Altera in describing their FPGA architectural characteristics to that of Xilinx. Consider things like (1) device naming, (2) logic clusters, (3) interconnect, (4) I/O, (5) DSP functionality, (6) memory. Limit your listing to FPGA-specific terms. You should be able to find at least 10 examples by going through the provided Altera documentation. Please be specific. *[Example from class: Xilinx refers to their clustering of LUTs as a Configurable Logic Block (CLB), whereas Altera refers to a similar clustering as a Logic Array Block (LAB). The fact that they both use the term "RAM" or "Adder" in their documentation would not count.]*

(b) Looking now at only the Xilinx Virtex-4 LX and the Altera Stratix II GX device families, describe how a large random logic function might appear differently on each. How about a large adder (requiring fast-carry logic)? How about a large multiplier using built-in multiplier resources? How about a large memory using built-in RAM resources?

All of the Xilinx and Altera documentation required to perform this analysis is available on the CprE 583 course web page: http://class.ece.iastate.edu/cpre583.

2) We discussed in class two different ways of representing numbers that have fractional components, and how these representations would affect the hardware implementation of the basic arithmetic operators (addition, multiplication, etc.).

(a) Examine both the Q9.23 fixed-point and IEEE 754 single-precision floating-point representations. What are the ranges of values (smallest number, largest number) that can be represented by each? We know that for any 32-bit binary representation, there are 2^{32} possible combinations of 0s and 1s. Given this fact, are there numbers that can be represented in IEEE 754 single-precision but not in Q9.23? Also, are there numbers that

can be represented in Q9.23 but not in IEEE 754? If your answer to either of these two questions is yes, please provide an example.

(b) Design an architecture (at the block diagram-level) that can perform addition or multiplication of two numbers represented in any arbitrary QI.F fixed-point notation. *[You can keep a practical limit to both I and F (64 or 128 would be fine), and use +, *, and mux operators as your building blocks. Worry more about how the input fixed-point numbers would have to be modified to be able to be added or multiplied together using standard binary adders and multipliers.]*

3) Most of the second week of class was spent on mapping various Boolean circuits to LUT-based computational structures. The following claim was made: a LUT of *k* inputs (*k*-LUT) can be programmed to implement any single-output logic function of *k* variables. The total number of possible logic functions / *k*-LUT truth table variations was calculated to be 2^2^*k*, or 65536 for *k*=4. It was also pointed out that many of these 2^2^*k* functions are redundant, as multiple truth tables can be obtained by just re-arranging the inputs of a single function. For example, given F(A,B) = ((not A) and B) and G(A, B) = ((not B) and A), than either F or G are not unique as F(A,B) = G(B,A).

(a) Design an algorithm that takes as its input *k* and calculates the number of non-redundant Boolean functions of *k* inputs. What would be the approximate run-time of your solution, as a function of *k*? What would be the approximate storage requirements of your solution, as a function of *k*? *[An exhaustive search algorithm is acceptable here.]*

(b) Implement the algorithm you came up with in part a) in any programming language of your choosing. Use your program to calculate the total number of unique LUT configurations for *k*=2, 3, 4. Attach your source file in a final .zip file to be included with your entire submission. *[I recommend being fairly confident in your solution to part a) before starting on this part. You should be able to manually determine the answer for the* k=2 *and* k=*3 cases in order to check your code. Based on what was discussed in class, the answer for* k=*4 should be on the order of (2*^2^k / k*!).]*

(c) Time how long it takes to obtain the answers in part b). Given these data points and the algorithmic analysis of part a), how long would you expect it to take to find all the unique *k*-LUT representations for *k*=5? For *k*=6? *[If C is your programming language of choice, you can reuse the gettimeofday() timing calls in the BubbleSort.c I provided in the first assignment. Many other timing analysis tools will give 0 time required for the* k=*2 and* k=*3 cases since it will be fairly quick.]*

4) See attached file *hw2.zip* which contains a partially-completed hardware implementation of the quadratic equation $Ax^2 + Bx + C$. Specifically, you should find the following files:

- *Adder.vhd* – a VHDL structural implementation of an addition operator, with output register.
- *Multiplier*.vhd – a VHDL structural implementation of a multiplication operator, with output register.
- *Quadratic.vhd* – a partial implementation of the quadratic equation, using the *Adder* and *Multiplier* modules described above. This design contains the instantiations of all the necessary adders and multipliers but they are not yet wired together.

- *altera/* – a sample Quartus II project directory for the *Quadratic* module *[not guaranteed to work with everyone's install of Quartus II.]*
- *xilinx/* – a sample ISE 8.1 project directory for the *Quadratic* module *[not guaranteed to work with everyone's install of ISE.]*

(a) Complete the implementation of the quadratic equation design. In a zip file, provide both the completed *Quadratic.vhd* file along with the Modelsim waveform file that shows correct values being computed for various inputs of variable *x*.

(b) Choose four FPGA devices to use for the following experiments. Two must be from Xilinx, and two from Altera. Choose at least 1 older device that does not have any built-in block multipliers. Provide a brief description of each of the FPGAs you've selected

(c) Synthesize the quadratic equation design onto each of your selected FPGAs, varying the input data width from {8, 16, 24, 32} bits by modifying the gDATA_WIDTH VHDL generic. Provide a chart detailing the area and performance characteristics of each of the designs on the four FPGAs. *[I'm looking for things like LUT/CLB usage, I/O usage, and estimated clock frequency. Both tools should provide a synthesis log file from which you can gather these numbers. The clock frequency after synthesis is only an estimate but it suffices for this purpose.]*

(d) There are four main components of the quadratic equation design: an adder, a constant adder, a multiplier, and a constant multiplier. For each of your selected FPGAs describe the RTL and technology schematic for the synthesized design of each component. How do these differ across devices? *[Check the Quartus II and ISE documentation for how to view these post-synthesis schematics. If it difficult to separate the components of the quadratic design in the schematic, you can synthesize them individually using the Adder.vhd and Multiplier.vhd files.]*

(e) Fully implement the design (place-and-route) on one Xilinx and one Altera FPGA. Describe the routed design schematic of each component in the design. How do these differ across devices? *[By implement here, I am referring to the Xilinx terminology. It's called "Fitter" in Altera Quartus II.]*