ComS / CprE 583 – Reconfigurable Computing

Take-Home Midterm Exam
Assigned: October 10
Due: October 12 (2:00pm)

[Note from Joe: As in the first three homework assignments, some of the questions here do not have a strictly correct answer. You will be graded on how well-formed your arguments are. Also, please keep in mind that you need to work individually on this and that no late submissions will be accepted.]

1) Review [25 points]

(a) Is an exhaustive set of pre-computed values (e.g. a sin/cos lookup-table) stored in memory an example of reconfigurable computing? Why or why not?

(b) In what scenarios might an anti-fuse FPGA be a more sensible choice than an SRAM FPGA?

(c) Why are FPGAs not able to run at 2-3GHz like conventional modern microprocessors?

(d) Why are most FPGA computational elements based on 4-LUTs? What would be a limitation of devices that use ($k < 4$) or ($k > 4$)?

(e) In what ways is the Xilinx XC4000 switchbox different from a universal switchbox? How could this affect practical designs?

(f) Under what conditions might one want to use distributed LUT-based memory resources as opposed to BlockRAM?

(g) What is a "slice"? How does it differ from a CLB?

(h) Apart from adders, describe the kind of circuits that are likely to make good use of the fast carry-chain logic found in commercial FPGAs.

(i) What is systolic computing, and how does it relate to reconfigurable computing? How are loop transformations utilized to implement systolic computing structures?

(j) How does the FPX architecture make use of two FPGAs? What advantages and limitations might there be of an equivalent single-FPGA approach?

(k) How does logic emulation differ from logic simulation? Why do logic emulators sometimes require multi-FPGA systems, while logic simulators can be run on standard PCs?

(l) Provide an example circuit where neither an ALAP nor ASAP scheduling will provide an optimal behavioral synthesis solution, in terms of required functional units.

2) Analysis [25 points]

(a) For memory-bound algorithms, a measure of *memory density* might be of greater importance than that of *computational density*. Argue for or against this concept. How do modern FPGAs compare to microprocessors in this regard? Extend this analysis to the concept of memory *bandwidth density*.

(b) Many of the LUT-based structures we have studied so far have had a single bit as output. Argue for or against the use of a *kc*-LUT, which has a *k*-bit input and computes *c* different Boolean functions.

(c) Consider a semi-constant multiplication operation, where the "constant" value is guaranteed to change every *N* iterations. How fast will the reprogramming operation need to be in order to justify the usage of a LUT-based constant multiplier, as compared to a standard array multiplier? Assume the multiplication is needed every iteration.

(d) Are systolic designs on average more area-efficient (in terms of throughput per slice) than functionally-equivalent iterative designs? Explain why or why not, considering the impact of clock frequency on throughput and efficiency.

(e) Will a partitioning-based algorithm be able to provide an optimal placement for an arbitrary circuit? If your answer is yes, briefly explain why. If your answer is no, please provide a counterexample.

3) Extension [50 points]

(a) Design an architecture that could be used to efficiently spell-check a large document using a SPLASH-like system. How would "close" matches be determined?

(b) Design a hybrid crossbar interconnect scheme for a multi-FPGA system, such that a two-dimensional mesh systolic structure can be implemented with single-hop delays for the East and West nearest neighbor connections, and three-hop delays for the North and South connections. For simplicity sake, assume that the horizontal dimension of your mesh will be five FPGAs or less. Can this scheme be easily modified to allow for a single-hop one-dimensional systolic array as well?

(c) Design a reconfigurable device which would be mainly used for computer architecture evaluation. What would be the base logic elements? How would they be arranged? What other design concerns would you have?