University of California at Berkeley
Department of Electrical Engineering and Computer Science
Reconfigurable Computing Seminar

CS294-7, Spring 1997 — **Administrivia** — Tuesday, January 21

Course Number:		CS294-7	
Course Name:		Reconfigurable Computing	
Term:		Spring 1997	
Control Number:		#24929	
Time:		TR 9:30-11AM	
Place:		Soda 505	
Instructors:		Dr. André DeHon	Prof. John Wawrzynek
	email	`amd@cs`	`johnw@cs`
	office	Soda 313	Soda 631
	hours	TR 1-3PM	
	phone	x3-2818	x3-9434
URL:		`<http://www.cs.berkeley.edu/~amd/CS294S97/>`	
Administrative Assistant:		Tim Ryan `<tryan@cs>`, Soda 634, x3-4014	

Description:

The emergence of high capacity reconfigurable devices is igniting a revolution in general-purpose processing. It is now becoming possible to tailor and dedicate functional units and interconnect to take advantage of application dependent dataflow. Furthermore, machines have been proposed that dynamically change their configuration with changing data sets and algorithm needs. Early research in this area of reconfigurable computing has shown encouraging results in a number of spot areas including cryptography, signal processing, and searching – achieving 10-100× computational density over more conventional processor solutions.

In this class we attempt to pull together the basic foundations of reconfigurable computing, sort out where it is interesting, and understand both how to architect reconfigurable systems and how to apply them to solving challenging computational problems. We will revisit basic questions about what building blocks we need in order to compute and rethink how we build computing systems based on today's technology costs.

Student Requirements and Grading:

1. Scribe notes for 2 or 3 lectures [25%]
2. Projects
   (a) Spatial Implementation of Computation [25%]
   (b) Research Oriented [50%]

(approximate weighting shown in brackets)

Scribe Notes

Primary Goal: Build a written reference for course material in the absence of a single reference text for the course. This provides everyone with a written reference to review during projects and after the course is completed.

- Prepare textual summary of lecture material in HTML.
- Each student will prepare notes for 2-3 lectures (depending on number of students in class).
- Send a draft to André within 3 business days of lecture. (*i.e.* by Friday afternoon for a Tuesday lecture, by Wednesday afternoon for a Friday lecture)
- André will suggest necessary revisions, if any.
- **Target:** have HTML version online for class a week after lecture is given.
- *N.B.* notes will be publicly accessible on the WWW.

Project 1: Implement a computational function in an FPGA.

- **Primary Goal:** Gain experience with spatial implementation of computations
- **Secondary Goals:**
  - Gain experience designing/using FPGAs with traditional design flow
  - Expand datapoints for FPGA/spatial implementation
- **Examples:**
  - priority queue (sorter)
  - median filter (RVF)
  - systolic GCD
  - sqrt
  - division ?
  - radix conversion ?

Project 2: Research project

- **Primary Goals:**
  - Demonstrate mastery of course material by (a) using in project, (b) describing project in context of course
  - Synthesize course material to explore new applications, archietctures, or techniques
- **Secondary Goals:**
  - Try out and evaluate new ideas
  - Answer open questions and quantify effects
  - Expand state of the art by studying new applications, architectures, analyses, and techniques
- **Ideas:**
  - *explore unexpected issues raised during project 1*
  - 3SAT solution (exploiting structural decompositions)
  - wire-routing/path-planning
  - solve a dynamic programming problem on FPGA
  - floating-point transcendentals, division
  - reconfigurable datastructures
  - espresso/eqntott – reimplement bitvector datastruct representation and operations and push to reconfigurable coprocessor
  - uncommon case handling (guard, detect, patchup)
  - runtime specialization
  - data/task depedent instruction-stream compression (for GARP? for XC4K)
  - measurement of Rent parameter for variety of structures
  - measure retiming requirements (to support highly pipelined operation)
  - Leiserson retiming for FPGA/DPGA (perhaps using Hyper do aggressive FPGA pipelining)
  - area-time (area-power) tradeoffs for common datapath operators
  - area-time (area-power) tradeoff for medium-grain subtask(s)
  - (stylized) Java→AFL prototype