

# "Teaching Reconfigurable Computing"

Experience With a Fundamentals Course

**Ron Sass**

**<http://www.rcs.uncc.edu/~rsass>**

University of North Carolina at Charlotte

June 1, 2008

# Outline

## A. What I Teach

# ECGR 4090/6090: Fundamentals of Reconfigurable Computing

- ▶ taught twice
  - ▶ Fall 2006: 24 graduate students finished the course
  - ▶ Fall 2007: 27 graduate + 1 undergraduate finished the course

# ECGR 4090/6090: Fundamentals of Reconfigurable Computing

- ▶ taught twice
  - ▶ Fall 2006: 24 graduate students finished the course
  - ▶ Fall 2007: 27 graduate + 1 undergraduate finished the course
- ▶ offered this summer:  $\approx$  17 graduate students enrolled
- ▶ will be offered next in Spring 2009

# Course Objectives

- ▶ "big picture" goal: students are able to
  - ▶ design special-purpose hardware cores
  - ▶ develop system software to interface cores and an application
  - ▶ assemble these hardware and software components into *computing systems*

# Course Objectives

- ▶ “big picture” goal: students are able to
  - ▶ design special-purpose hardware cores
  - ▶ develop system software to interface cores and an application
  - ▶ assemble these hardware and software components into *computing systems*
- ▶ specific skills
  - ▶ understand FPGA device organization
  - ▶ tool competency
  - ▶ performance trade-offs
  - ▶ business economics

# Agenda

- ▶ for me the (not so) hidden agenda:  
*create a large pool of students with the skills to be effective  
in my research lab*

# Agenda

- ▶ for me the (not so) hidden agenda:  
*create a large pool of students with the skills to be effective in my research lab*
- ▶ collateral student benefit:  
*these are very marketable job skills*

# Format

- ▶ twenty-eight 75-minute sessions
  - ▶ 18/28 lectures
  - ▶ 9/28 interactive/demo
  - ▶ 1/28 testing (mid-term)
- ▶ a session is interleaved: 1/3 lecture, 1/3 demo, 1/3 lecture
- ▶ six labs
- ▶ around 9-12 short (10 min) quizzes

# Lectures

- ▶ just a giant information dump:
  - ▶ here are seven different buses, their characteristics, and when to use one versus another
  - ▶ this is how a LUT is implemented CMOS and note its power characteristics
  - ▶ these are configuration options required to compile Linux for an FPGA
- ▶ in short: here's a bunch of facts you need to know

# Lectures

- ▶ just a giant information dump:
  - ▶ here are seven different buses, their characteristics, and when to use one versus another
  - ▶ this is how a LUT is implemented CMOS and note its power characteristics
  - ▶ these are configuration options required to compile Linux for an FPGA
- ▶ in short: here's a bunch of facts you need to know
- ▶ not the most effective use of instruction time!
- ▶ but ... we currently lack any central resource (like a *textbook* to teach from!)

# Labs

- ▶ about 70 developer boards in the lab (ML-310/ML-410)
- ▶ largely tutorial
  - ▶ type this...
  - ▶ observe this...
  - ▶ type this...
  - ▶ observe this...
  - ▶ type this...
  - ▶ observe this...
  - ▶ **now extend this base to do something simple**
- ▶ meant to breed familiarity (exposure to basic tool flow)
- ▶ does not require analytical skills

## Labs (cont'd)

- ▶ Lab 1: Hello, World
- ▶ Lab 2: Adding a Peripheral
- ▶ Lab 3: Compiling and Running Linux
- ▶ Lab 4: Linux Device Driver
- ▶ Lab 5: Hardware Standalone Device (CRC)
- ▶ Lab 6: Linux Device Driver with HW CRC

# Quizzes

- ▶ quick-and-to-the-point (two 5-point questions)
- ▶ focus on day-to-day material
  - ▶ simple recitation
  - ▶ some are challenging: feedback on rank among peers
  - ▶ designed to prep students for exams
  - ▶ question to initiation discussion
- ▶ always graded but small percentage of overall grade

# Content

post-mortem analysis on Fall 2007 course yields:

- ▶ 7/28: FPGA device characteristics
- ▶ 1/28: FPGA systems organization (PCI bus, co-processor, etc.)
- ▶ 5/28: vendor-specific tools (EDK/XPS)
- ▶ 15/28: system software (cross-compiling, Linux, root fs)
- ▶ 2/28: system design
- ▶ 1/28: graph theory
- ▶ 1/28: hardware/software partitioning

## B. What I Wish I Taught

# What's Missing?

after the basics, students need to learn graph-theoretic solutions to

- ▶ spatial computation: parallelism and timing
- ▶ pipelining: maximizing throughput
- ▶ profiling and hardware/software partitioning
- ▶ hardware/software interfacing

## How To Get From A to B

# Warning!

- ▶ brainstorming ideas ... untested
- ▶ topics for discussion

# Some Ideas

- ▶ Off-Line Resources
  - ▶ wiki tutorials (KU's microblaze site)
  - ▶ textbook would be nice (assigned readings)
- ▶ Remote Access — tools to make the lab more accessible
- ▶ Stable Tools — progress is great but difficult to track!