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

CS294-7, Spring 1997 | Project 2: Assignment | Tuesday, March 10

Project 2: Research project in Reconfigurable Computing

Dates

Assigned:	March 11
Proposals:	March 18 (HTML proposal pointer to `amd@cs` by class time)
Due:	May 6 (HTML report pointer to `amd@cs` by class time)
Talks:	May 6 and 8

Goals

- **Primary Goals:**
  - Demonstrate mastery of course material by: (a) using in project, (b) describing project in context of course (place project motivation, studies, and results within the issues discussed and frameworks developed during course)
  - Synthesize course material to explore new applications, architectures, 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
  - Spawn longer term research

Scope

This is intended to be a decent sized project which you dig into for 6-8 weeks. The nature of the project is fairly open. Most projects probably fall into one (or more) of the following broad categories:

- Applications
- Tools
- Architectures
- Measurements and Analysis

During the project you should be able to do a full implementation of a non-trivial instance of some application, prototype a mapping or programming tool, estimate resource requirements across a benchmark set, or obtain initial results on the merits of an architectural feature.

Good(?) Ideas

There are a number of "good" ideas which we have floating around as things to consider for the next generation of devices, mapping tools, and programming strategies. These are all good fodder for consideration directly (*e.g.* architectural studies) or indirectly (*e.g.* as you study application X, how would some of these architectural modifications or programming styles effect the efficiency of the application?). Of course, you are welcome to help generate your own "good" ideas, too.

N.B. There is quite a bit more detail behind the points bulletted here—see us for details.

- **Architecture**
  - Sea-of-gates interconnect strategy (Day 12)
  - Support for relocatable macro blocks
    - separate routing resources for inter and intra macro routing
    - repositionable (perhaps w/in hierarchical constraints) w/out rerouting (re-generating different configuration bits)
  - Input retiming registers (this came up on Day 2, a bit in the context of highly pipelined systolic designs, and will come up again when we talk about retiming on Days 22-24)
  - Multicontext ($c > 1$ in our model)
  - Processor and array integration
  - Configuration edits and compression (compressing instruction storage and bandwidth requirements – recall on Day 3 we saw that instruction storage and bandwidth requirements could end up dominating other resources)
  - Memory integration
  - Cascade support (minimize the path length for latency limited designs)
  - Dynamic run-time management of array resources
- **Programming**
  - Parameterized Library generator scheme
  - Fast (on the fly) composition of data-specialized applications
  - Stylized Java programming strategy
  - Synchronous Data Flow

- **General Themes**
  - Use silicon efficiently → high-throughput implementations (interleaved tasks, if necessary) – *e.g.* applications running at 100-200MHz on today's FPGA technology
  - Specialization (specialize implementations around early bound data)
  - Implement typical case in array with trap out to processor for odd cases
  - Fast compilation (or assembly) of array configurations

Benchmarks

As a young field, there is not a well-defined set of "benchmarks" applications in a suitable form for use. Consequently, we'll have a bit of work to pick and choose computations, kernels, and applications to use along with our studies. Some obvious staring points include:

- **Circuit/CAD Benchmarks**
  - MCNC logic, fsm
  - DSP Synth ?
- **DSP Benchmarks**
- **Image and Audio Processing Applications**
  - Intel Image Processing Benchmarks
  - Image Understanding Benchmarks (?)
- **Implementations from Project 1**
  - I would like to pool up the implementations from project one as the beginnings of a more "computationally" oriented benchmark set (to complement the random logic / FSM orientation we see in a traditional logic benchmark suite)
- **"Old" applications from the Reconfigurable Computing literature**
  - FIR
  - ATR
  - Sequence Matching
  - DES
  - RSA encoding

Resources

Have or can definitely get:

- Xilinx Tools from Project 1
- Xilinx 6200 Tools (Beta release)
- Various free software tools (UCLA, Toronto, ...)
- Hardware
  - XC4K development boards
  - Metalithic – Digital Wings for Audio

Could likely get (if these are interesting to anyone, we should start checking on availability soon):

- Altera tools
- Hardware
  - Altera RC board
  - Giga Ops Spectrum board
  - VCC XC6200 board (3/14?)

Project Suggestions

Following these themes, here's a list of potential projects:

- Calculate Rent parameter across a large collection (diversity) of circuits and computations
- Investigate sea-of-gates FPGA using Toronto, symmetric FPGA model (VPR and SEGA?).
- Calculate retiming requirements to achieve fully pipelined circuits and computations (try to formulate a retiming rule as a complement to Rent's rule?)
- Area-Time (power) tradeoffs for common FPGA library operators
- Design-dependent compressibility of library operators and/or particular applications
- Intel image processing benchmarks on a processor with integrated reconfigurable array
- 3SAT solution (exploiting structural decompositions)
- `espresso/eqntott` – reimplement bitvector datastruct representation and operations and push to reconfigurable coprocessor
- Leiserson retiming on FPGA (solve shortest path problems)
- Dynamic scheduling for stream processing applications

- Fast (run-time) macro generator/assembler for specialized instantiation on a processor-coupled array (and/or dynamic sizing/partitioning)
- Generator language
- Stylized Java $\rightarrow$ RC array compiler
- Direct FSM $\rightarrow$ multicontext FPGA mapping
- Fast mapping w/ multicontext routing
- Layout and compare Universal v/s Xilinx switch boxes
- Layout M choose N input connection block; compare encoded v/s decoded input select
- Quantify specialization benefits on applications
- Audio stream processing (?)
- Accelerator for FPGA PPR (?)
- Dynamic data size determination (?)

Project Proposal

Your project proposal should clearly describe your project including:

- goals (what do you want to understand?)
- scope (what do you have to do?)
- strategy (how will you attack it?)
- milestones and options (how will you pace your project? what are your options if some pieces turn out to be more time involved than you originally anticipated?)
- evaluation (how will you evaluate your tool/architecture/implementation/characteristics)

This should be 500-1000 words and include diagrams and references as necessary to clarify your project plan. Send a pointer for an HTML document with your proposal to `amd@cs.berkeley.edu` by lecture time on March 18th.

We will review the proposals to give you early feedback on your chosen project and strategy. Since the range of these projects is much broader than project 1, we will plan on meeting with everyone during the week following the project proposal due date. We're currently planning to have people sign up for a 25 minute timeslot in the window 1-4PM Tuesday, March 18 or 12-4PM Wednesday, March 19. Sign up for slots during class on Thursday (March 13).

Report

For your report, prepare an HTML page describing your project. This should be about 3000 words and include adequate diagrams, graphs, and tables to convey your results. You should relate your task, problems, and findings to the issues and discussion covered in class. Your paper report should be self-contained, including:

- motivation
- concepts
- details of your construction experiments, and benchmarks
- results
- lessons or recommendations

In Class Project Talks

We have set aside the final two class periods for everyone to present their results as a capstone for the course. We will probably have to schedule the talks for a slightly broader time period than just class time to accommodate everyone. Plan for a 15 minute slot – 10-12 minutes of presentation with 3-5 minutes for questions.