CS250
VLSI Systems Design

Fall 2020

John Wawrzynek

with

Arya Reais-Parsi
Reconfigurable Array Design Research

Many of these topics have been studied, but remain open (or secret).

As with most design decisions, it comes down to design space exploration and cost/benefit analysis. Ideally, finding the Pareto optimal frontier.

In FPGA design, it is complicated by the fact that decisions are interrelated. For instance, CLB design strongly affects interconnection needs.

- **Architecture**
  - **Configurable Logic Blocks**
    - Alternative to LUTs (less area, delay)

The Actel ACT architecture

![Figure 28 - Quicklogic (Cypress) Logic Cell.](image)
Reconfigurable Array Research

- Architecture
  - Configurable Logic Blocks
    - LUT size:


“Finally, our results show that a LUT size of 4 to 6 and cluster size of between 3-10 provides the best area-delay product for an FPGA”
Reconfigurable Array Research

- Architecture
  - Configurable Logic Blocks
    - Internal interconnection among LUTs/FFs


“we show that mapping to a 7-input LUT structure can approach the performance of 6-input LUTs while retaining the area and static power advantage of 4-input LUTs”

Figure 1. Hardwired and soft-wired S44s
Reconfigurable Array Research

- Architecture
  - Configurable Logic Blocks
    - ALU versus LUT based, processor cores
  - Hybrid fine-grained / coarse grained?


Reconfigurable Array Research

- Architecture
  - Interconnection Network (relatively unexplored)
    - CLOS network, Fat-trees, other ad hoc topologies
    - Tool place and route time critical
  - Configuration Structure
    - Partial reconfiguration granularity
    - dynamic reconfiguration (reconfigure while running)
    - multiple context
    - debugging interface
Reconfigurable Array Research

- Implementation
  - Standard Cells versus Full Custom


- Hybrid

Reconfigurable Array Research

- Implementation
  - Full custom fabric layout generation (process portability, rapid DSE):
    - Sticks

- BAG - Berkeley Analog Generator

Reconfigurable Array Research

- The “RISC” of FPGAs
- **Reduced Complexity Reconfigurable Array (RCRA)**
  - Compared to state-of-art commercial arrays, can a very simple array compete with (or beat) on:
    - Performance (clock frequency), area efficiency, power efficiency,
    - area/delay product, power/delay product?
  - Do the PPR (partition, place, & route) tools speed up?
- Ideas:
  - If local interconnect is efficient, then perhaps clustering is not necessary.
  - Are LUTs overkill?
  - Simpler interconnect?
## Class Project Options

**Do we want to do architecture research or simply focus on implementation issues?**

**Do you want to do research in implementation issues?**

- Reimplement existing array: Xilinx xc2064, xc4000
  - documented architecture with tool support
    - more pedagogically interesting then as research project
- New Design:
  - first attempt at RCRA: simple, fast, area efficient, low-pwr
  - multi-context optimized for dynamic reconfiguration
  - hybrid coarse-grain fine-grain array
  - simpler is better for class projects
- How many teams?
- How to divide up functionality and/or be redundant?
Define Fabric Architecture

Collect Benchmarks

Develop RTL Simulation

Adapt PPR Tools

Sweep Parameters

Component Gate/Transistor Design

CLB | Interconnect | Config | RAM | ALU

Verify with Unit Tests/Fabric-Simulator

Build Fabric Layout Generator

Layout of Components

Fabric Layout

Fabric Physical Verification
Teams

- How many people per team? 1, 2, 3, ...

- Possible Division of Labor:
  - By Component: CLB, Interconnect, Configuration, Hard-blocks
    - high-level design, simulation models, layout, testing
  - Plus Integration Team
    - manages interfaces, fabric level simulator framework, fabric layout generator

- Redundant Teams
  - CLB variations, or standard cell versus full custom
  - Interconnect variations, ...

New topic: Question - Verilog or Chisel?
Short Assignment - mail us

- If you have a partner already, who they are.
- What you would like to work on.
- Some details of what’s in your part or how you would go about figuring it out.

We will provide:

- feedback on suitability of idea (technical and load balancing project needs)
- Match making
- ask for a presentation to make to class

Due Tuesday next week by class time.

- Feedback by Wednesday
- Short Presentations the following week
Upcoming Lectures

- **Outline**
  - Architecture (done FPGAs, others?)
  - Methodology/Tools
  - Circuits

- Assume you are a system designer wanting to build a chip.
  - Not a researcher in process/device development, or CAD tool developer, or ...
  - This assumption informs our choice of topics

- **Your job:** *mapping function to structure.*
  - Unlike in software, we don’t have good enough tools (yet) to do it automatically.
Digital Design

Given a functional description and performance, cost, & power constraints, come up with an appropriate implementation using a set of primitives.

- How do we learn how to do this?

1. Learn micro-architectural templates
2. Learn about primitives
3. Learn about design representations and tools
   Map translate between representations
   Transform for optimization (move towards or along the Pareto Frontier)
4. Use trial and error - CAD tools and prototyping. Practice!

- Digital design is in some ways more an art than a science. The creative spirit is critical in combining primitive elements & other components in new ways to achieve a desired function.

- However, unlike art, we have objective measures of a design:
  
  **Performance  Cost  Power**
Typical ASIC Implementation Path

- Behavioral Model (C++, python, ...)
- HDL represents RTL
  - Behavioral specification
  - Structural
  - Instantiation of special blocks - “black boxes” (ex: RAM, ROM)
- Library cells from the PDK (standard cells)
- GDSII is the modern standard file format for representing layout information (layers and geometry)
HDL to ASIC layout flow

“push-button” approach

module adder64 (a, b, sum);
    input [63:0] a, b;
    output [63:0] sum;
    assign sum = a + b;
endmodule
Standard cell layout

- With limited # metal layers, dedicated routing channels were needed
- Currently area dominated by wires

1um, 2-metal process

Modern sub-100nm process
“Transistors are free things that fit under wires”
Standard cell design

Layout considerations

Cells have standard height but vary in width
Designed to connect power, ground, and wells by abutment
Macro (Black Box) modules

- Generate highly regular structures (entire memories, multipliers, etc.) with a few lines of code
- Verilog models for memories automatically generated based on size

256×32 (or 8192 bit) SRAM Generated by hard-macro module generator
End of Lecture 7