What is Reconfigurable Computing?

• Computation using *hardware* that can *adapt* at the logic level to solve *specific* problems

° *Why is this interesting?*
  
  • Many applications are poorly suited to microprocessors
  
  • VLSI “explosion” provides increasing resources. How can we use them?
    
    - “field-programmable” devices
  
  • Allows for high performance, bug fixes, and fast time-to market for a selection of applications
Background needed for this course

1. Basic VLSI – transistors, delay models.

2. Basic algorithms – graph algorithms, searches

3. Computer Architecture – ALU, microprocessor

4. Digital Design – adder, counter, etc.

Topic self-contained!

Reconfigurable Computing is a lot more than just devices
Course Organization

• Homework assignments (15%)

• Final project (30%)

• Mid-term (25%)

• Final exam (25%)

• Class participation/attendance (5%)
  - Students expected to participate in class discussion

• No required text – readings will be assigned from research papers.
  - We will use material from the three survey papers on the course website a lot
What characterizes Reconfigurable Computing?

Parallelism, specialization, hardware-level adaptation

- Parallelism customized to meet design objectives
- Logic specialized to perform specific function
- Functionality changed as problem requirements change
Microprocessor-based Systems (Temporal)

- Generalized to perform many functions well.
- Operates on fixed data sizes.
- Inherently sequential
  - Constrained even with multiple data paths.
Reconfigurable Computing

• Create specialized hardware for each application.
• Functional units optimized to perform a special task.

if (A > B) {
    H = A;
    L = B;
} else {
    H = B;
    L = A;
}
Example: Bubblesort (Spatial)

- Adapt interconnect to problem.
- Take advantage of parallelism.
Implementation Spectrum

- ASIC gives high performance at cost of inflexibility.
- Processor is very flexible but not tuned to the application.
- Reconfigurable hardware is a nice compromise.
Reconfigurable Hardware

A \land B \land C \land D = \text{out}

- Each LUT operates on four one-bit inputs.
- Output is one data bit.
- Can perform any boolean function of four inputs

\[ 2^4 = 64K \text{ functions (4096 patterns)} \]
Logic Element

Inputs

Look-Up Table (LUT)

Clock

Enable

State

Out
Field-Programmable Gate Array

- Each logic element outputs one data bit.
- Interconnect programmable between elements.
- Interconnect tracks grouped into channels.
FPGA Architecture Issues

- Need to explore architectural issues.
- How much functionality should go in a logic element?
- How many routing tracks per channel?
- Switch “population”? 

Logic Element
In the Beginning - Xilinx XC4000 Cell

Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

- 2 4-input look-up tables
- 1 3-input look-up table
- 2 D flip flops
In the Beginning - Xilinx XC4000 Routing

XC4000E and XC4000X Series Field Programmable Gate Arrays

Lecture 1: Course Introduction

September 6, 2016
Altera Stratix Logic Array Blocks (Clusters)
Routing Connections

Based on the switch and wire parasitic, interconnect routes can be modeled as \( RC \) networks.

Other issues:
- Power
- Routability
Design abstractions

1. English specification
2. Executable program behavior
3. Sequential machines register-transfer logic
4. Logic gates, transistors circuit layout
5. Throughput, design time Function units, clock cycles
6. Literals, logic depth nanoseconds
7. microns
High-level Compilers

- Difficult to estimate hardware resources.
- Some parts of program more appropriate for processor (hardware/software codesign).
- Compiler must **parallelize** computation across many resources.
- Engineers like to write in C rather than pushing little blocks around.

```c
for (i = 0; i < n, i++)
{
    c[i] = a[i] + b[i]
}
```

Some success stories
Translating a Design to an FPGA

\[ C = A + B \]

- CAD to translate circuit from text description to physical implementation well understood.
- Most current FPGA designers use register-transfer level specification (allocation and scheduling)
- Same basic steps as ASIC design.
Circuit Compilation

1. Technology Mapping

Assign a logical LUT to a physical location.

2. Placement

Select wire segments and switches for interconnection.

3. Routing
Technology Mapping: A Simple Example

Made of Full Adders

\[ A + B = D \]

Logic synthesis tool reduces circuit to SOP form

\[ S = \overline{ABC}_i + \overline{ABC}_i + \overline{ABC}_i + ABC_i \]

\[ C_o = \overline{ABC}_i + \overline{ABC}_i + \overline{ABC}_i + ABC_i \]
Lecture 1: Course Introduction

Processor + FPGA

Three possibilities

1. FPGA serves as coprocessor for data intensive applications – possible project.

2. FPGA serves as embedded computer for low latency transfer.

"Reconfigurable Functional Unit"
3. Processor integration

- FPGA logic embedded inside processor.
- A number of problems with 2 and 3.
  - Process technology an issue.
  - ALU much faster than FPGA generally.
  - FPGA much faster than the entire processor.
Programmable System on a Chip

- Specialized hardware is commonly used for compute-intensive applications
  - Coprocessors (FPU, graphics, sound, …)
  - Accelerator boards (FPGA boards, I/O cards, …)
- FPGAs also perform well for many of these apps
- Reconfigurable logic in System-On-A-Chip
Dynamic Reconfiguration

- What if I want to exchange part of the design in the device with another piece?
- Need to create architectures and software to incrementally change designs.
- Effectively a “configuration cache”

Examples: encryption, filtering.
Classifying Reconfiguration

Reconfiguration methodology
- Static
- Partially static (=partial reconfiguration)
- Dynamic
Types of Reconfiguration

° Types of reconfiguration
  • Static reconfiguration – application is not running
  • Semi-static – different portions of application *time-sliced* on same hardware fabric
  • Dynamic reconfiguration – application modified in response to changing environmental issues

° What is reconfigured?
  • Processor reconfiguration – customized instruction sets, pipelining, bit width
  • Parameter reconfiguration
  • Control reconfiguration
  *Can you think of other applications?*
Hot Reconfigurable Computing Research Areas

• Determining what fixed components should go in an FPGA (I/O, routers, network interfaces, etc)

• Important new applications for reconfigurable devices (especially embedded applications and security)

• Better understanding the role of standard microprocessors, GPUs, and reconfigurable hardware.
  - They all have benefits
  - Programming environments are improving
  - Computing in the cloud

• Compiler / run-time environments
  - Broadening of programming (Altera: OpenCL, Xilinx: Vivado)
  - Operating system management / virtual machine
Course Software: Versatile Place and Route

- Performs FPGA placement and routing.
- Written in C
- Runs on Linux and PCs
  - We support execution on quark.ecs.umass.edu
  - Generally straightforward to install and use.
- Estimates device sizes and performance
- Very widely used in FPGA research community
- More advanced version of VPR are available.
  - Appropriate for use in course projects
Ideas for the Course Project

- Evaluate an application using a microprocessor and an FPGA
  - Consider performance and power consumption

- Modify/update CAD algorithms associated with VPR

- Model modern FPGAs which include multiple dice.

- FPGA security: encryption, hardware obfuscation

- High-level compilation
  - Adapt and evaluate LegUp (high-level synthesis tool)
  - Use OpenCL compiler to target multiple platforms

*Topic must involve experimentation.*
Summary

° Reconfigurable computing relies heavily on new VLSI technology

° Device architectures have matured.
  • New features constantly considered

° Application development progressing at rapid pace

° Integration of hardware and software a difficult challenge

° FPGAs are a multi-billion dollar a year business
  • Devices are much different now than 20 years ago

° Active area of research at UMass.