ESE534: Computer Organization

Day 10: February 15, 2012
Instruction Space

Previously

- Temporally Programmable Architectures
- Spatially Programmable Architectures
- Instructions

Today

- Instructions
  - Requirements
  - Taxonomy

Computing Requirements (review)

Instruction Control

- What needs to be controlled per ALU-memory-bank?

Instruction Taxonomy
Instructions

- Distinguishing feature of programmable architectures:
  - Instructions -- bits which tell the device how to behave

Focus on Instructions

- Instruction organization has a large effect on:
  - size or compactness of an architecture
  - realm of efficient utilization for an architecture

Terminology

- Primitive Instruction ($pinst$)
  - Collection of bits that tell a single bit-processing element what to do on each cycle
  - Includes:
    - select compute operation
    - input sources in space
      - (interconnect)
    - input sources in time
      - (retiming)

Preclass

- How big is $pinst$ for preclass?
  - (problem 2)

Computational Array Model

- Collection of computing elements
  - compute operator
  - local storage/retiming
  - Interconnect
  - Instruction

“Ideal” Instruction Control

- Issue a new instruction to every computational bit operator on every cycle
“Ideal” Instruction Distribution

• Why don’t we do this?

Preclass

• How many total instruction bits?
  – (preclass 3)

• How many pins on a chip?

“Ideal” Instruction Distribution

• Problem: Instruction bandwidth (and storage area) quickly dominates everything else
  – Compute Block ~ 256K F^2 (512F x 512F)
  – Instruction ~ 64 bits
  – Wire Pitch ~ 4F
  – Memory bit ~ 300F^2

Preclass

• How wide is instruction distribution? (F units)

• For F=32nm?

Instruction Distribution

How many instructions
Across side??

Two instructions in 512F
64x4F = 256F

Instruction Distribution

Distribute from both sides = 2x
Instruction Distribution

Distribute X and Y = 2x

Instruction Distribution

- Room to distribute 2 instructions across PE per metal layer (512F = 2x64x4F)
- Feed top and bottom (left and right) = 2x
- Two complete metal layers = 2x
- How many instructions per PE side?
  ⇒ 8 instructions / PE Side

Instruction Distribution

- Maximum of 8 instructions per PE side
- When saturate wire channels?
- Saturate wire channels at 8x√N = N
- What N?
  ⇒ at 64 PE
  - beyond this:
    * instruction distribution dominates area

Instruction Distribution

- Perimeter = 4x2√N ≤ N
- Saturate wire channels at 8x√N = N
  ⇒ at 64 PE
  - beyond this:
    * instruction distribution dominates area
    * instruction consumption goes with area
    * Instruction bandwidth goes with perimeter

Avoid Instruction BW Saturation?

• How might we avoid this?

• Beyond 64 PE, instruction bandwidth dictates PE size

\[
\frac{\sqrt{\text{PE}_{\text{area}} \times 4x\sqrt{N}}}{(64x4F)} = N
\]

PE_{\text{area}} = 4KF^2xN

• As we build larger arrays
  ⇒ processing elements become less dense
Instruction Memory Requirements

- **Idea:** put instruction memory in array
- **Problem:** Instruction memory can quickly dominate area, too
  - Memory Area = 64 × 300F²/instruction
  - PE_area = 256K F² + (Instructions) × 20K F²

When instruction memory dominate?

Instruction Pragmatics

- Instruction requirements *could* dominate array size.
- Standard architecture trick:
  - Look for structure to exploit in "typical computations"

Typical Structure?

- What structure do we usually expect?

One Extreme

- SIMD (Single Instruction Multiple Data)
  - e.g. microprocessors, GPUs
  - Instruction/cycle
  - share instruction across array of PEs
  - uniform operation in space
  - operation variance in time

Another Extreme

- FPGA (Field-Programmable Gate Array)
  - Instruction/PE
  - assume temporal locality of instructions (same)
  - operation variance in space
  - uniform operations in time
Spatially Programmable

Network with Configuration

VLIW

- VLIW = Very Long Instruction Word
  - Few \textit{pins} / cycle
  - Share instruction across \( w \) bits

Architectural Differences

- What differentiates a VLIW from a multicore?
  - \textit{E.g.}
    - 4-issue VLIW vs.
    - 4 single-issue processors

SIMD, Spatial, VLIW, Multicore

Basis Vectors

- In practice, mix together:
  - \textit{E.g.} Modern Multicore
    - MIMD (multiple cores, one PC per core)
    - VLIW within core (superscalar, multiple \textit{pins} issue from each core)
    - Word-wide SIMD for Integer operations
      - Perhaps even with explicit SIMD operations
Placing Architectures

- What programmable architectures (organizations) are you familiar with?

Gross Parameters

- Instruction sharing width
  - SIMD width
  - granularity
- Instruction depth
  - Instructions stored locally per compute element
- pinsts per control thread
  - E.g. VLIW width

Architecture Taxonomy

<table>
<thead>
<tr>
<th>PCs</th>
<th>Pints/PC</th>
<th>depth</th>
<th>width</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
<td>1</td>
<td>1</td>
<td>FPGA</td>
</tr>
<tr>
<td>1</td>
<td>N (48,640)</td>
<td>8</td>
<td>1</td>
<td>Tabula ABAX (A1EC04)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>32</td>
<td>Scalar Processor (RISC)</td>
</tr>
<tr>
<td>1</td>
<td>N</td>
<td>D</td>
<td>W</td>
<td>VLIW (superscalar)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Small</td>
<td>W/N</td>
<td>SIMD, GPU, Vector</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
<td>D</td>
<td>W</td>
<td>MIMD</td>
</tr>
<tr>
<td>16</td>
<td>1 (47)</td>
<td>2048</td>
<td>64</td>
<td>16-core</td>
</tr>
</tbody>
</table>

Instruction Message

- Architectures fall out of:
  - general model too expensive
  - structure exists in common problems
  - exploit structure to reduce resource requirements
- Architectures can be viewed in a unified design space

Big Ideas

- Basic elements of a programmable computation
  - Compute
  - Interconnect
    - (space and time, outside system [IO])
    - Instructions
- Instruction resources can be significant
  - dominant/limiting resource

Admin

- Reading on blackboard
- HW5
  - Problem 1 due Monday
    - Should be able to do all of Problem 1 now
  - Day11/Monday relevant to Problem 2
- Class Monday
- No class next Wednesday