ESE534: Computer Organization

Day 10: February 15, 2012
Instruction Space

Previously

- Temporally Programmable Architectures
- Spatially Programmable Architectures
- Instructions

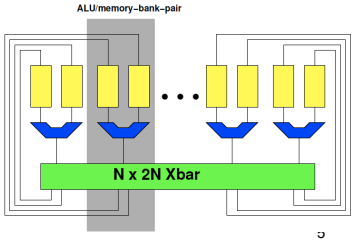
Today

- Instructions
  - Requirements
  - Taxonomy

Computing Requirements (review)

Instruction Control

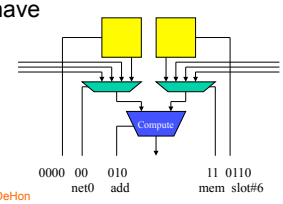
- What needs to be controlled per ALU-memory-bank?



Instruction Taxonomy

Instructions

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

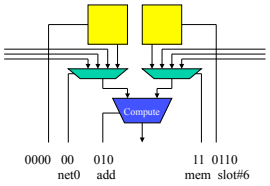


Focus on Instructions

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

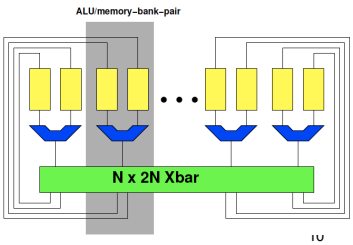
Terminology

- **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**)



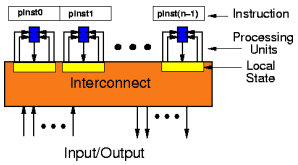
Preclass

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



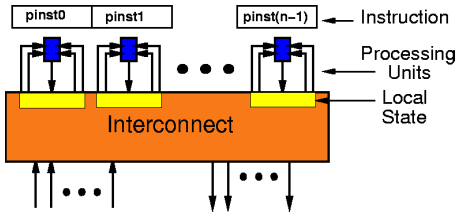
Computational Array Model

- 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?

Preclass

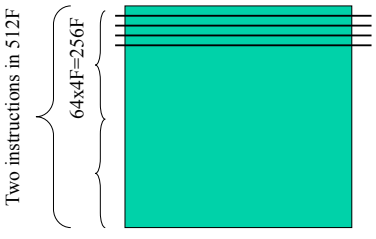
- How wide is instruction distribution? (F units)
- For F=32nm?

"Ideal" Instruction Distribution

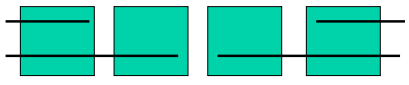
- **Problem:** Instruction bandwidth (and storage area) quickly dominates everything else
  - Compute Block ~ 256K F^2 (512F x 512F)
  - Instruction ~ 64 bits
  - Wire Pitch ~ 4F
  - Memory bit ~ $300F^2$

Instruction Distribution

How many instructions Across side??

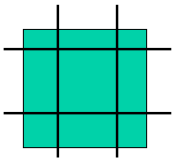


Instruction Distribution



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 = 2×64×4F)
- Feed top and bottom (left and right) = 2×
- Two complete metal layers = 2×
- 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 $8\times\sqrt{N} = N$
- What N?
- ⇒ at 64 PE
  - beyond this:
    - instruction distribution dominates area

Instruction Distribution

- Perimeter = $4\times2\sqrt{N} \le N$
- Saturate wire channels at $8\times\sqrt{N} = N$
- ⇒ at 64 PE
  - beyond this:
    - instruction distribution dominates area
- Instruction consumption goes with area
- Instruction bandwidth goes with perimeter

Instruction Distribution

- Beyond 64 PE, instruction bandwidth dictates PE size

How PE_{area} grow? $\frac{\sqrt{PE_{area}}\times4\times\sqrt{N}}{(64\times4F)} = N$

$PE_{area} = 4KF^2\times N$

- As we build larger arrays

  ⇒processing elements become less dense

Avoid Instruction BW Saturation?

- How might we avoid this?

Instruction Memory Requirements

- **Idea:** put instruction memory in array
- **Problem:** Instruction memory can quickly dominate area, too
  - Memory Area = $64 \times 300F^2$/instruction
  - $PE_{area} = 256K\ F^2 + (\text{Instructions}) \times 20K\ F^2$

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?

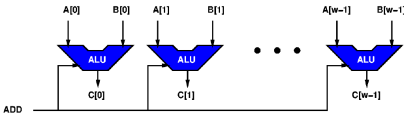
- What structure do we usually expect?

One Extreme

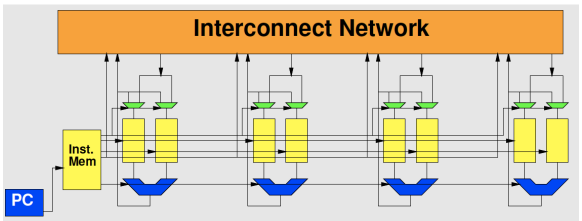
- SIMD (Single Instruction Multiple Data)

e.g. microprocessors,GPUs

- Instruction/cycle
- share instruction across array of PEs
- uniform operation in space
- operation variance in time

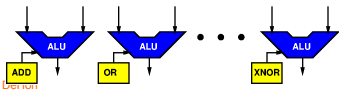


SIMD

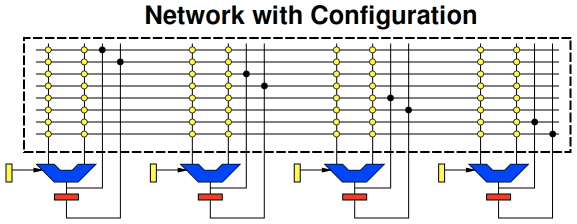


Another Extreme

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

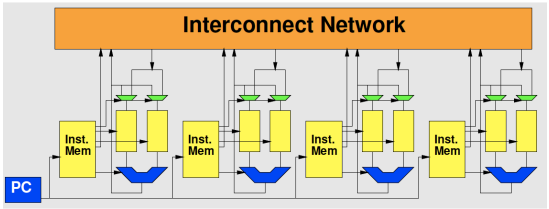


Spatially Programmable



VLIW

- VLIW = Very Long Instruction Word
  - Few *pinsts*/cycle
  - Share instruction across *w* bits

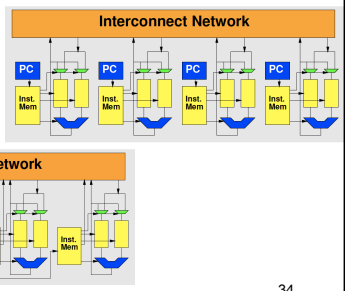


Architectural Differences

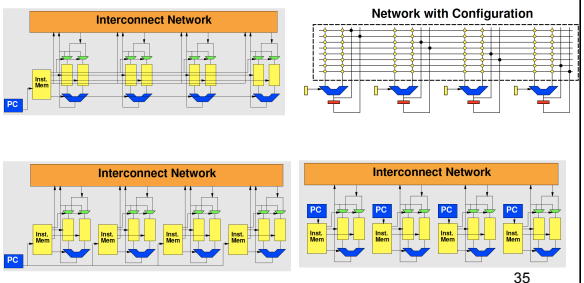
- What differentiates a VLIW from a multicore?
  - *E.g.*
    - 4-issue VLIW vs.
    - 4 single-issue processors

Architectural Differences

- What differentiates a VLIW from a multicore?



SIMD, Spatial, VLIW, Multicore



Basis Vectors

- In practice, mix together:
  - E.g. Modern Multicore
    - MIMD (multiple cores, one PC per core)
    - VLIW within core (superscalar, multiple pinst 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

PCs	Pints/PC	depth	width	Architecture
0	N	1	1	FPGA
1	N (48,640)	8	1	Tabula ABAX (A1EC04)
1	1	1024	32	Scalar Processor (RISC)
1	N	D	W	VLIW (superscalar)
1	1	Small	W*N	SIMD, GPU, Vector
N	1	D	W	MIMD
16	1 (4?)	2048	64	16-core

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

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

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