# ESE680-002 (ESE534): Computer Organization

Day 8: February 5, 2007

Computing Requirements and Instruction Space




## Previously

- Fixed and Programmable Computation
- Area-Time-Energy Tradeoffs
- VLSI Scaling



## Today

- Computing Requirements
- Instructions
  - Requirements
  - Taxonomy



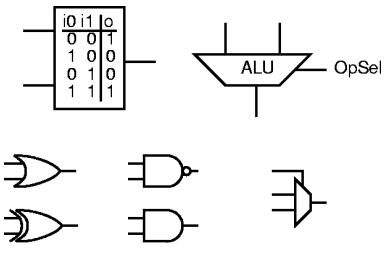
## Computing Requirements (review)



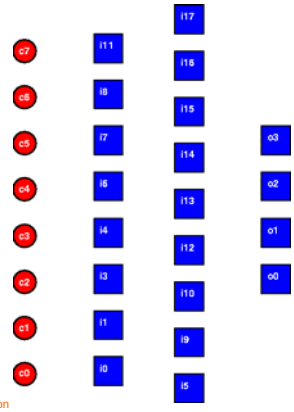
## Requirements

- In order to build a **general-purpose** *(programmable)* computing device, we absolutely must have?
  - _
  - _
  - _
  - _
  - _

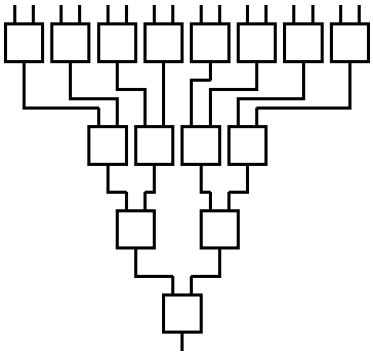




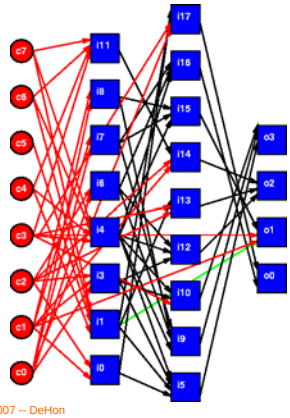
Primitive compute elements enough?





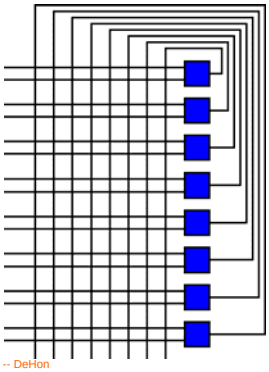







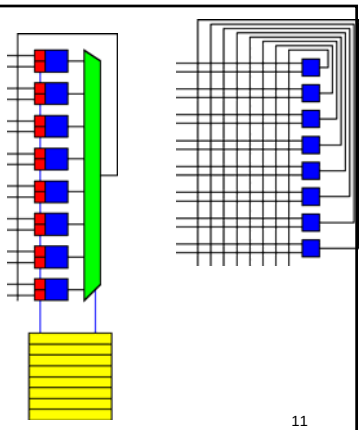



Compute and Interconnect



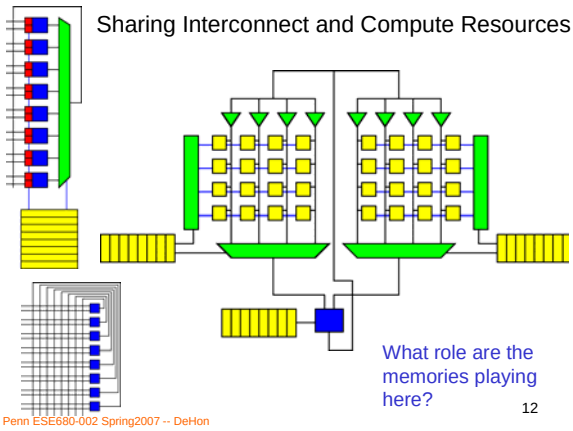


Sharing Interconnect Resources





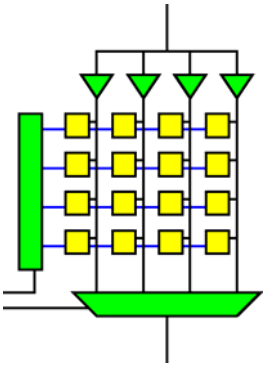
Sharing Interconnect and Compute Resources



What role are the memories playing here?

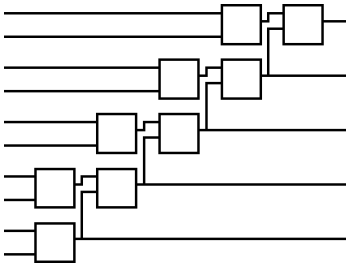
Memory block or Register File

**Interconnect:** moves data from input to storage cell; or from storage cell to output.

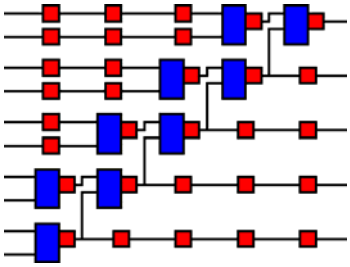




What do I need to be able to use this circuit properly? (reuse it on different data?)









# Requirements

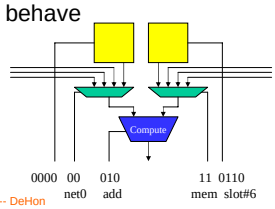
- In order to build a **general-purpose** *(programmable)* computing device, we absolutely must have?
  - Compute elements
  - Interconnect: space
  - Interconnect: time (retiming)
  - Interconnect: external (IO)
  - Instructions



# Instruction Taxonomy



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



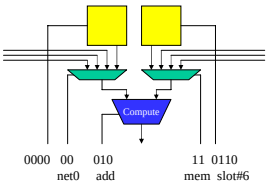
## Focus on Instructions

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



## Terminology

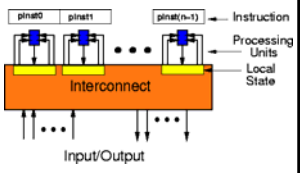
- **Primitive Instruction (*pinst*)**
  - Collection of bits which tell a single bit-processing element what to do
  - Includes:
    - select **compute** operation
    - input sources in space
      - (**interconnect**)
    - input sources in time
      - (**retiming**)






## Computational Array Model

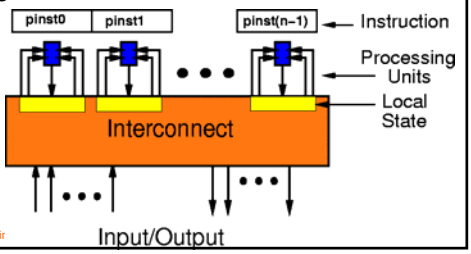
- Collection of computing elements
  - compute operator
  - local storage/retiming
- Interconnect
- Instruction






## "Ideal" Instruction Control

- Issue a new instruction to every computational bit operator on every cycle






## "Ideal" Instruction Distribution

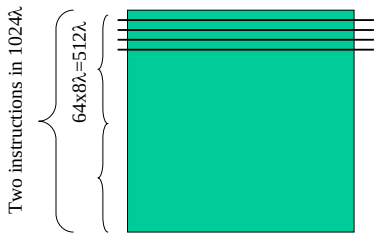
- Why don't we do this?



## "Ideal" Instruction Distribution

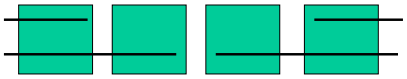
- **Problem:** Instruction bandwidth (and storage area) quickly dominates everything else
  - Compute Block ~ $1M\lambda^2$ ($1K\lambda$ x $1K\lambda$)
  - Instruction ~ 64 bits
  - Wire Pitch ~ $8\lambda$
  - Memory bit ~ $1.2K\lambda^2$

## Instruction Distribution





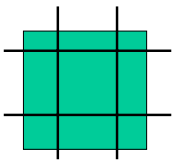

## Instruction Distribution



Distribute from both sides = 2x



## Instruction Distribution



Distribute X and Y = 2x



## Instruction Distribution

- Room to distribute 2 instructions across PE per metal layer (1024 = 2×8×64)
- Feed top and bottom (left and right) = 2×
- Two complete metal layers = 2×

- ⇒ 8 instructions / PE Side



## Instruction Distribution

- Maximum of 8 instructions per PE side
- 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

$$\frac{\sqrt{PE_{area}}\times 4\times\sqrt{N}}{(64\times 8\lambda)} = N$$

$PE_{area} = 16K\lambda^2\times N$

- 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 = $64 \times 1.2K\lambda^2$/instruction
  - $PE_{area} = 1M\lambda^2 + (\text{Instructions}) \times 80K\lambda^2$



## Instruction Pragmatics

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



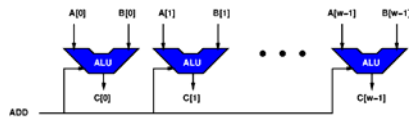
## Typical Structure?

- What structure do we usually expect?



## Two Extremes

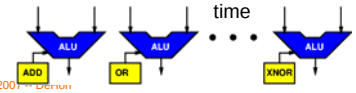
- SIMD Array (microprocessors)
  - Instruction/cycle
  - share instruction across array of PEs
  - uniform operation in space
  - operation variance in time






## Two Extremes

- SIMD Array (microprocessors)
  - Instruction/cycle
  - share instruction across array of PEs
  - uniform operation in space
  - operation variance in time
- FPGA
  - Instruction/PE
  - assume temporal locality of instructions (same)
  - operation variance in space
  - uniform operations in time



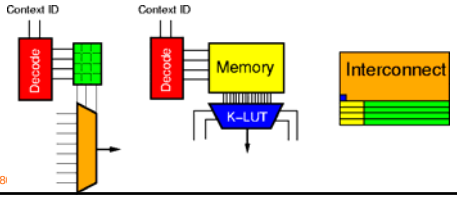
## Placing Architectures

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



## Hybrids

- VLIW (SuperScalar)
  - Few *pinsts*/cycle
  - Share instruction across *w* bits
- DPGA
  - Small instruction store / PE






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



## 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 Instruction Taxonomy

| Control Threads (PCs) | *pinsts* per Control Thread | Instruction Depth | Granularity | Architecture/Examples |
|---|---|---|---|---|
| 0 | 0 | 0 | n/a | Hardwired Functional Unit (*e.g.* ECC/EDC Unit, FP MPY) |
| | $n$ | 1 | $w$ | Reconfigurable ALUs |
| 1 | 1 | $c$ | $w$ | Traditional Processors |
| | | | $n_r \cdot 1$ | Bitwise SIMD |
| | | | $n_r \cdot w$ | Vector Processors |
| | $n$ | $c$ | 1 | DPGA |
| | | 8 | 16 | PADDI |
| | | $c$ | $w$ | VLIW |
| $m$ | $n$ | 1 | 1 | HSRA/SCORE |
| | 1 | $c$ | $n_r \cdot w$ | MSIMD |
| | $m$ | $c$ | 1 | VEGA |
| | | 8 | 16 | PADDI-2 |
| | | $c$ | $w$ | MIMD (traditional) |



## 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

- Instruction assignment due Wednesday
- Reading for today and Wed. on web
- Try GRW262 for André Office Hours this week



## Big Ideas [MSB Ideas]

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



## Big Ideas [MSB-1 Ideas]

- Two key functions of memory
  - retiming
  - instructions
    - description of computation