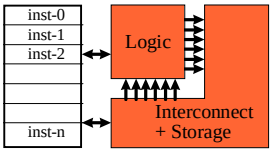
Reconfigurable Computing Intro
January 14, 1998

- Admin
- Architecture Overview
- Cal
- RC Design Space
- RC Research Agenda

Admin

- Newsgroups are up
  cmu.cs.class.cs828.{announce,discuss}
- Lab1 - start now
- /afs/cs/academic/class/15828/bin/checkin
  cklog cs.cmu.edu
- **`handin # file`**
- Schedule Verilog tutorial
- Wait list

An Abstract Architecture

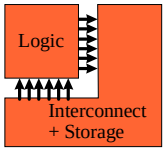


Three components to all computing devices:

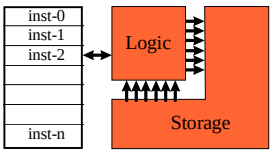
- Control Store
- Compute Elements
- Communication

Hardware

- No Control Store
- Not General Purpose
- All "computing" is through spatial connections.



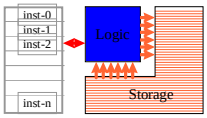
Traditional MPU



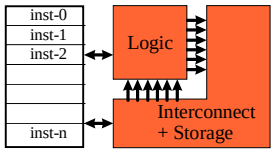
- Control store only controls logic
- Communication is in time
  Through registers, memory, etc.

Variations on MPU

- Control Store/Instruction Distribution
  - Scalar: one-one
  - SIMD: one-many
  - VLIW: many-one
  - MIMD: many-many
- Logic
  - Superscalar, RISC, CISC, Vector
- Storage/Communication
  - SMP, DMP, Distributed

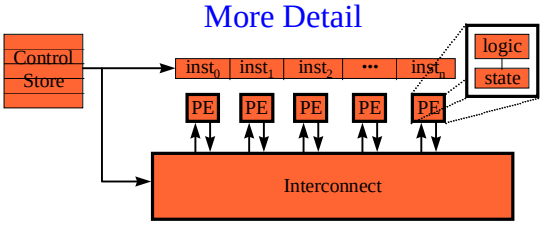


Reconfigurable Computing



- Control Store for both compute and communicate
- Communicate in both space and time.

More Detail



In Abstract:

Instruction configures both PE and interconnect every cycle.

In Reality:

Instruction Bandwidth/Memory too high, so …

Compromise

RC Compromises

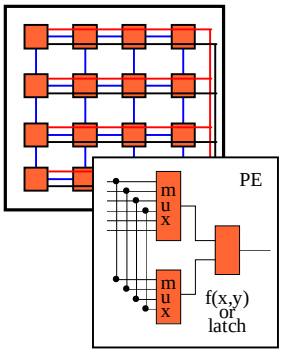
Reduce requirements by

- Encoding instruction

  I.e., reducing programmability of device
- Reducing frequency of instruction issue.
- Both in practice

Choices relate to granularity of device and instruction bandwidth that is delivered to the device.

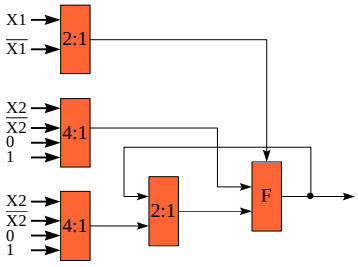
The Cal Chip



- 32x32 array of single bit PEs
- Each PE
  - Can compute f(x,y)
  - Or, store state
  - Two global lines
- NEWS connections
- Bit serial programming

  > 2^{14} bit instruction word

The CAL function block



Cal Architecture

- Homogenous architecture
- Fine-grained PEs
- Local Interconnect
- PEs used for routing
- Obeys rents rule internally? Externally?
- Programmed by bit-serial link! >2^{14} bits needed
- Can be partially reconfigured

An extreme point in the design space.

Rent's Rule

Empirical relationship between logic and interconnect

$w = Kg^p$

w = number of wires
g = # of gates
p < 1.0

Typically, $0.5 \leq p \leq .8$

Captures notion of locality in circuits

RC Design Space

- Granularity
- Logic Block Architecture
- Interconnect
- Programmability
- System Interface
- Application

Tradeoffs optimize for
- instruction bandwidth
- compute density

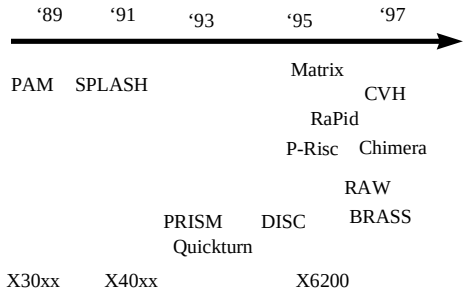
Interconnect

- Direct Connect or Wire Channels
- Mix of wires
  - short, medium, long lines
- Switches
- Internal Interconnect
- Routing delay

System Interface

- Attached Processor
- Co-Processor
- Function Unit
- Stand-alone

History



Issues in RC Research[Buell]

- Are FPGAs large enough?
- How much of architecture should be exposed to the programmer?
- Programming Language? Compilers?
- Range of architectures needed?
- How do CCMs fit in between ASIC and GPU?

Are FPGAs large enough?

- Partitioning is a big problem in terms of
  - compilation time
  - I/O wires (Rents rule)
  - performance
  - performance cliff

Virtualize!

Exposing the Architecture

Today's tools require programmer to be aware of:

- system level architecture
- fabric level architecture
  - PE
  - interconnect
  - routing delays

Need an abstract machine model.

Programming Language/Compiler

- Expressibility
  - Hardware Description Language?
  - How to expose parallelism?
  - Partition between CCM target and Host?
- How "compile" when no machine model?

How do we compile for the new abstract model?

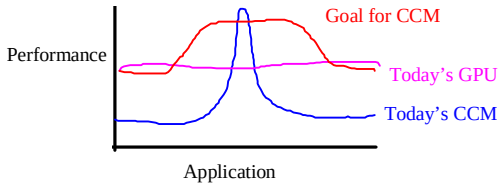
Range of architecture needed?

- Are these devices limited-purpose or general-purpose?
- Can we redefine general-purpose?

Need a benchmark suite.

Where do CCMs fit in ASIC-GP Spectrum?

- FPGAs are certainly in between.
- Are RC systems in between?



Research Issues

- Architectures
  - System Interface
  - Control/Instruction Distribution
  - Interconnect
  - Model/Metrics
- Compiler
  - Generality
  - Speed
  - Partitioning
  - Algorithms
- Applications