Reconfigurable Computing Seminar
January 15, 1998

- What is RC?
- Motivating RC
- Why Study RC Now?
- This Class

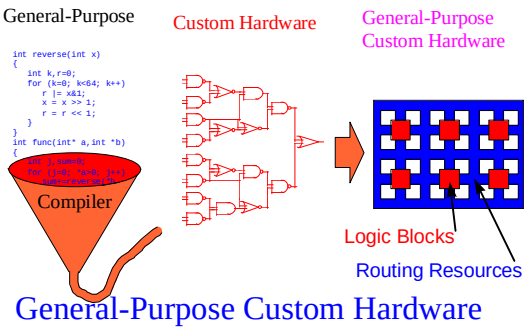
Why Reconfigurable Computing?

- Performance (~ 10x Speedup)

Algorithm	FPGA system	Comparison CPU	Speedup
DNA Matching	SPLASH 2	SPARC 10	4300
RSA Crypto	PAM	Alpha 150MHz	17.8
Ray Casting	RIPP-10	Pentium 75MHz	33.8
FIR filter	1 Xilinx FPGA	DSP 50MHz	17.9
Hidden Markov Model	1 Xilinx FPGA	SPARC 10	24.4
Spec92	MIPS+RC	MIPS	1.12

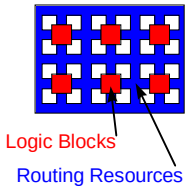
- Efficiency (~ 10x) In term of Bit-ops/λ^2s
- Lower Chip Costs (~ .5x)
  - Increased yield
  - Decreased complexity
  - Decreased design cost

What is RC?



General-Purpose Custom Hardware

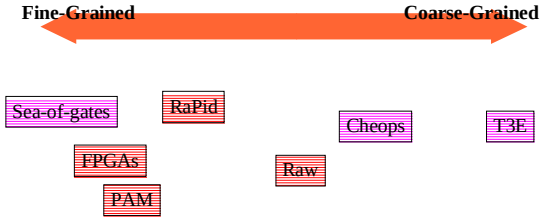
What is RC?



My Answer:

General-Purpose Custom Hardware

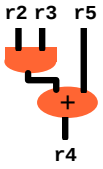
Spectrum of Reconfigurable Computers



Our emphasis will be on fine-grained compute elements. E.g., T3D is not a RC.

Another Definition of RC

A reconfigurable computer is a device which computes by using post-fabrication spatial connections of compute elements. [Dehon]



Compare this to General Purpose Computers which compute by making connections in time.

```
and r1,r2,r3
add r4,r1,r5
```

Why Reconfigurable Computing?

- Performance (~ 10x Speedup)

Algorithm	FPGA system	Comparison CPU	Speedup
DNA Matching	SPLASH 2	SPARC 10	4300
RSA Crypto	PAM	Alpha 150MHz	17.8
Ray Casting	RIPP-10	Pentium 75MHz	33.8
FIR filter	1 Xilinx FPGA	DSP 50MHz	17.9
Hidden Markov Model	1 Xilinx FPGA	SPARC 10	24.4
Spec92	MIPS+RC	MIPS	1.12

- Efficiency (~ 10x) In term of Bit-ops/λ^2s
- Lower Chip Costs (~ .5x)
  - Increased yield
  - Decreased complexity
  - Decreased design cost

Market for RC

- General Purpose Computing?
- Embedded Computing
  - Market bigger than GP in all metrics
    - # of devices
    - Dollar amount
  - Very Demanding
    - power
    - area
    - performance
- Telecom
- Automotive
- Settop boxes
- Space
- Etc.

Why not build it once?

- Changes in application requirements
- Standards can be a moving target
- time-to-market
- Design costs
- Production costs
- Flexibility
- Sometimes even better performance

E.g., 56K modem market

Benefits to RC

- Increased compute density

  More active area dedicated to computing and less to control, instruction decoding, etc..
- Match compute to data word size
- More parallelism
  - Instruction level
  - Bit level
  - Pipeline
  - Task
- Increased fault-tolerance

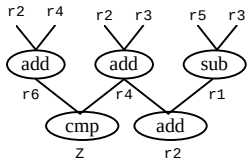
Kinds of Parallelism

- MIMD (application level)
- SIMD (application loop level)
- ILP (Instruction level)
- Pipeline (loop level)
- Bit-level

MPPs use only one level of parallelism

Instruction Level Parallelism

```
add   r2,r4,r6
add   r2,r3,r4
sub   r5,r3,r1
add   r4,r1,r2
cmp   r6,r4
```

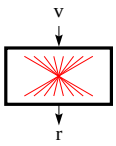


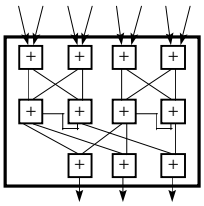
- Superscalar must find dataflow graph at run time
- RC constructs dataflow graph at compile time.
  - No FU limitations
  - No control logic overhead
  - No window size limitations

Bit-Level Parallelism

```
Reverse(int v) {
  int x. r;

  for (x=0; x<WIDTH; x++) {
    r |= v&1;
    v = v >> 1;
    r = r << 1;
  }
  return r;
}
```

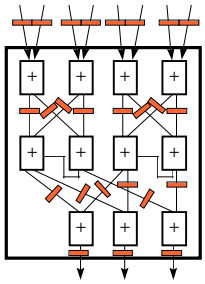
```
popcount(int v) {
  int r=0;

  while (v) {
    if (v&1) r++;
    v = v >> 1;
  }
  return r;
}
```





Pipeline Parallelism

```
for (j=0; j<MAX; j++)
   b[j] = popcount(a[j]);
```



Application of RC

- Application Attributes
  - Highly Regular
  - Pipeline-able
  - Variable bit-width
- Three Sample Applications
  - Mobile Helicopter
  - Satellite
  - Genetic Programming

Why Study RC NOW?

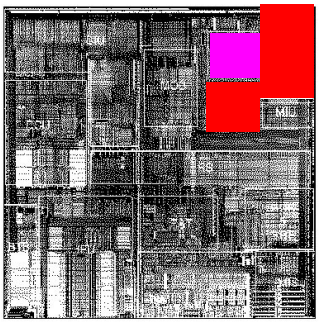
- Increasing # of transistors
- Complexity and cost of chip designs
- FPGAs coming into their own (the 1k law)
- Current computing demands are RC friendly

  Desktop demands driven NOT by excel or word, but by multimedia, encryption, filters, ...

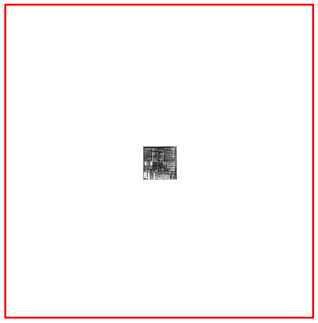
A very good time to be doing RC research.

Industry can't do it: Brains over Brawn.

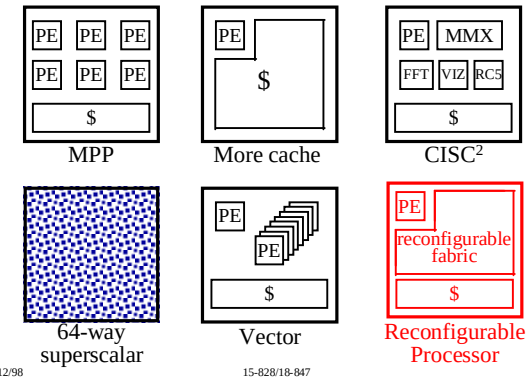
Processor of Today



In 10 Years!



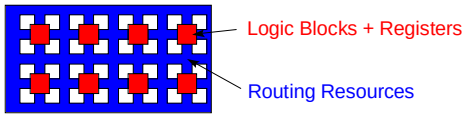
Using the Silicon



Challenges to RC

- Better tools
  - There is currently no way to program RC systems in a general purpose manner
  - Current tools are
    - slow
    - hardware oriented
- Better architectures
  - FPGA is current basis and it is not suited for RC
  - Floating point not really possible

Field-Programmable Gate Arrays Are Basis For RC



✔Infinitely reconfigurable
✗ Oriented toward “glue” logic
✗ “Programmed” by hardware design
✗ Configuration time is *very* long

Goals

- Understand Reconfigurable Computing
- Generate Original Research

Understanding RC

- Foundations of RC
  - FPGAs
  - CAD (Place and Route, Technology Mapping)
  - Architecture (Logic Blocks, Interconnect)
  - Compilation
- State of the Art

Goals of Class

- Attack one or more of the hurdles in RC.
- Present to local community
- Publish!

The Course

Approx

- Meet twice a week (1:30-3:00 WeH 5409B)
- Read and review assigned papers (~2)
- Scribe classes (~2 per student)
- Presentations

40%

- Two labs — 20%
- One original research project. — 40%

Topics - I

- FPGA Basics
  - Logic Block Architecture
  - Interconnect
- CAD/Compilation
  - Technology Mapping
  - Placement and Routing
  - HW/SW Partitioning
  - Partial Evaluation
  - Programming Languages

Topics - II

- Reconfigurable Architectures
  - History
  - Systolic Machines
  - Functional Unit Approaches
  - Partial Reconfiguration Machines
- Applications

Admin

- Labs
- ECE student accounts
- Setting up directories
- Handing in work
- Grades

http://www.cs.cmu.edu/afs/academic/class/15828-s98/www

Immediate Attention

- Run `/afs/cs/academic/class/15828/bin/checkin`
- Papers for Wednesday
- Lab1