The CMU Reconfigurable Computing Project

April 9, 1999

Mihai Budiu

mihaib@cs.cmu.edu

Current Project Members

CS Department

Seth Copen Goldstein
Mihai Budiu

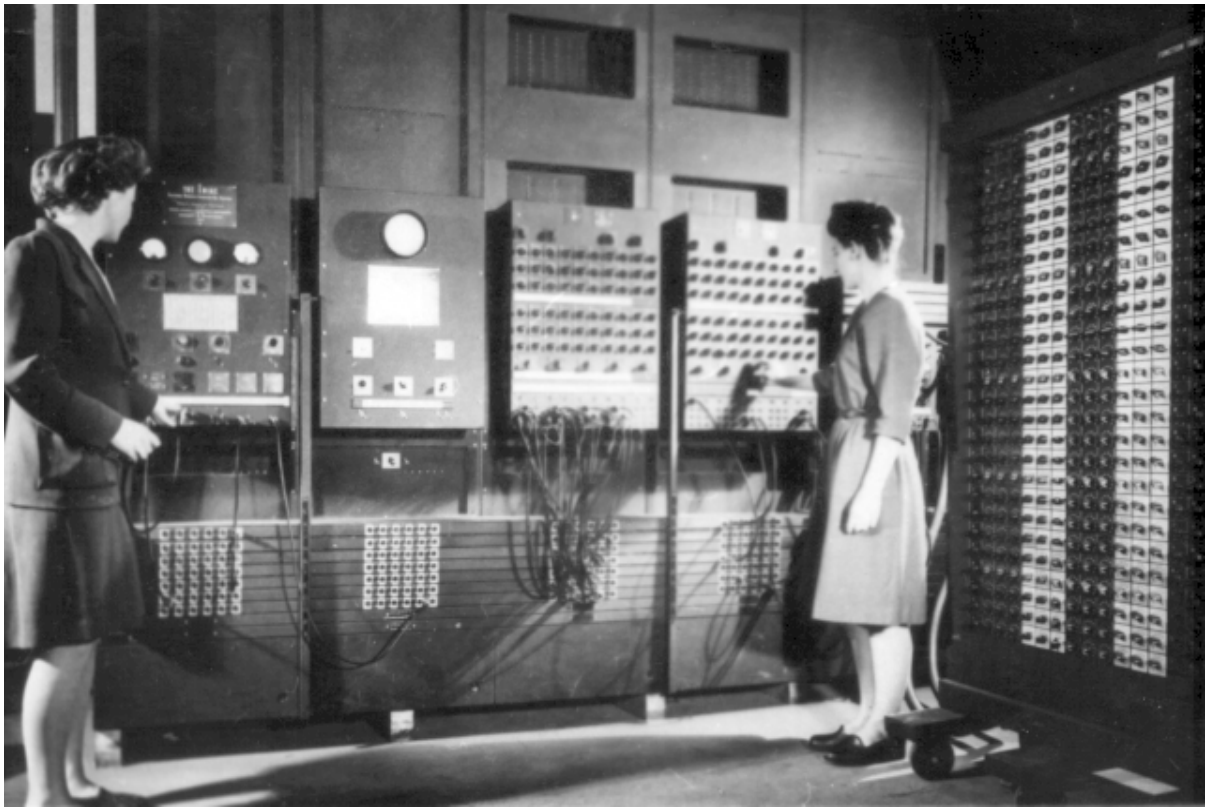
ECE Department

Herman Schmit
Srihari Cadambi
Matt Moe
Robert Taylor
Ronald Laufer

Why Study Reconfigurable Hardware?

It is a nice computation paradigm

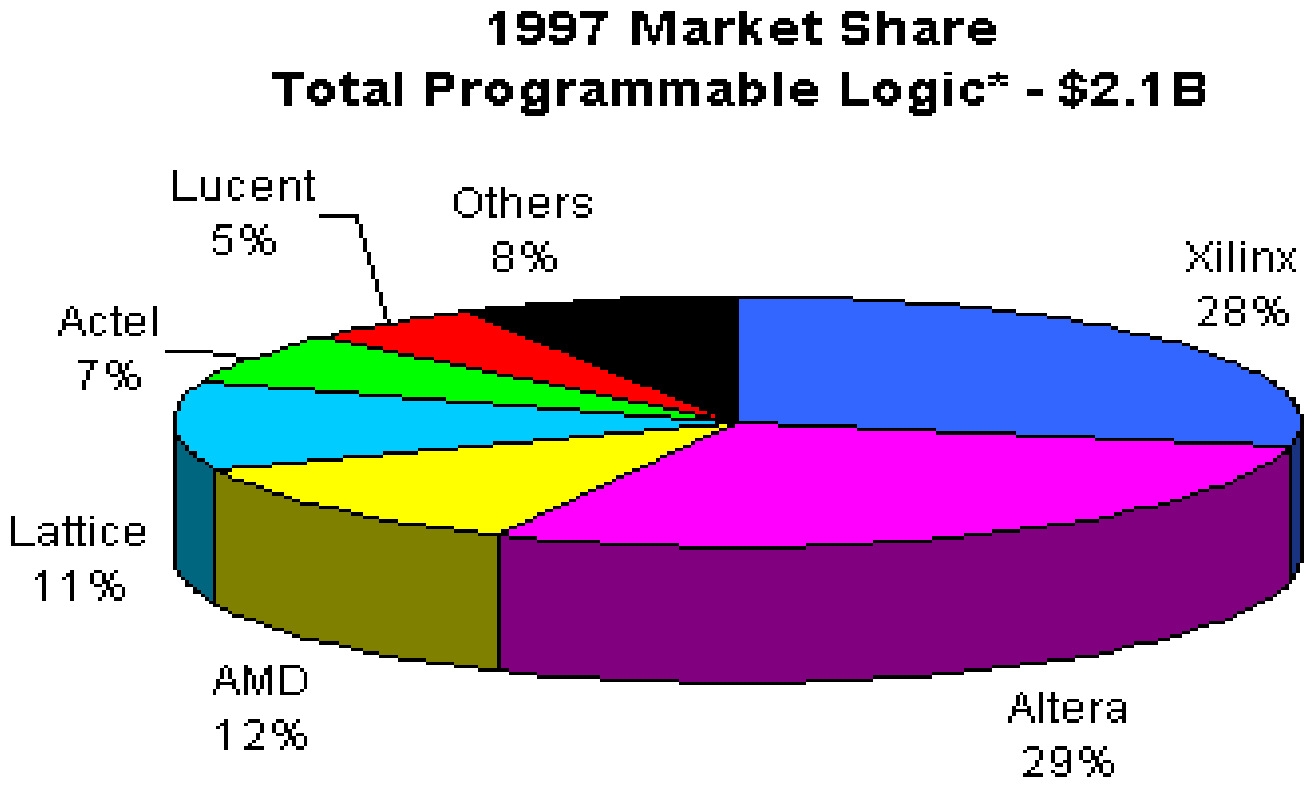
(wire your own computer)



Why Study Reconfigurable Hardware

Algorithm	Year	System	Versus	Speedup x
DNA matching	1992	SPLASH 2	SPARC 10	4300
FIR Filter	1998	PipeRench	UltraSparc 300Mhz	90
IDEA Encryption	1998	PipeRench	UltraSparc 300Mhz	61
SAT solver	1997	Pamette	SPARC 5 110Mhz	17--1100
Ray Casting	1995	RIPP-10	Pentium 75Mhz	33.8
Hidden Markov Model	1996	1 Xilinx FPGA	SPARC 10	24.4
DES Encryption	1996	GARP	UltraSparc 170Mhz	24
SPEC92	1994	MIPS+RC	MIPS	1.22

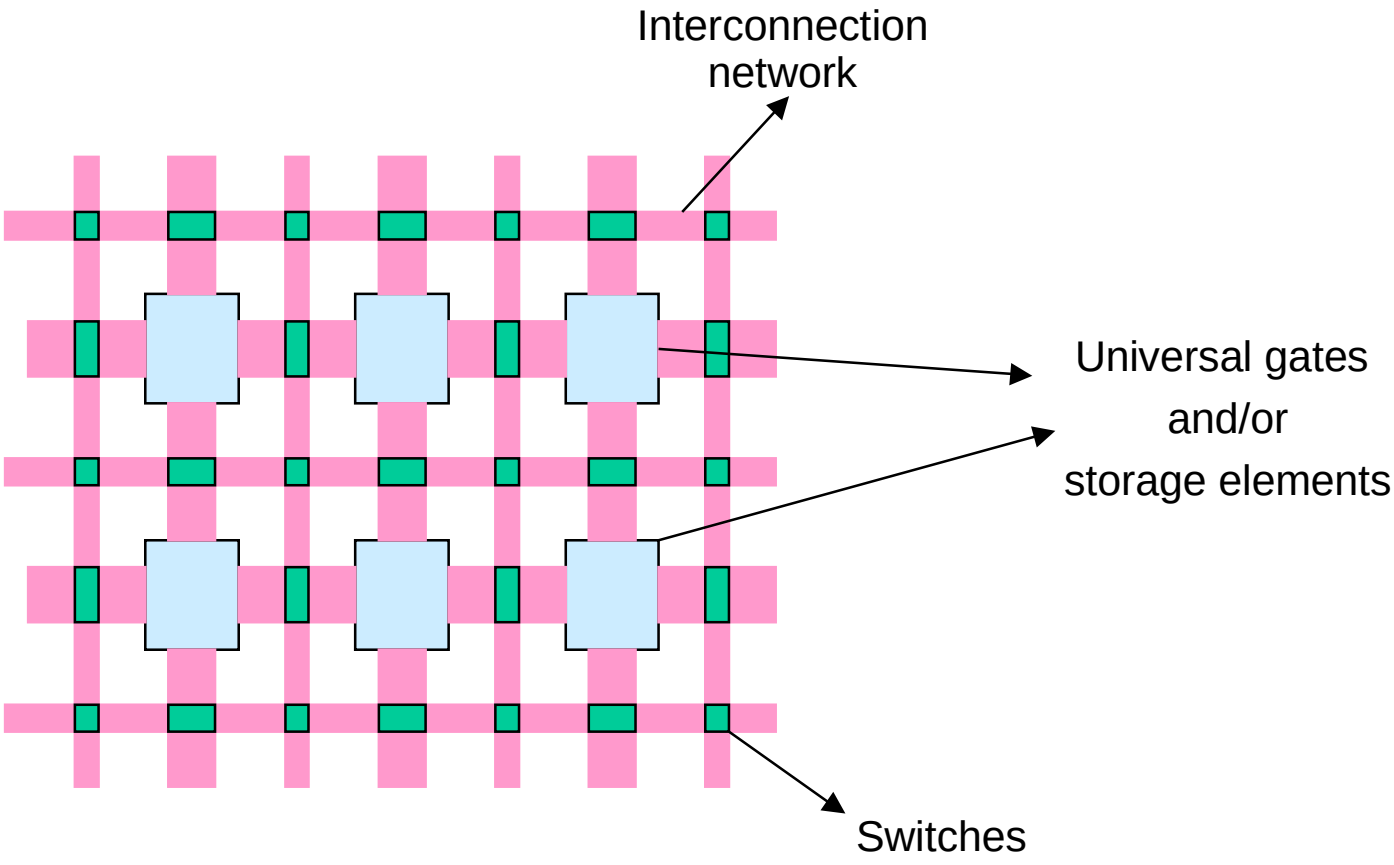
Commercial Players



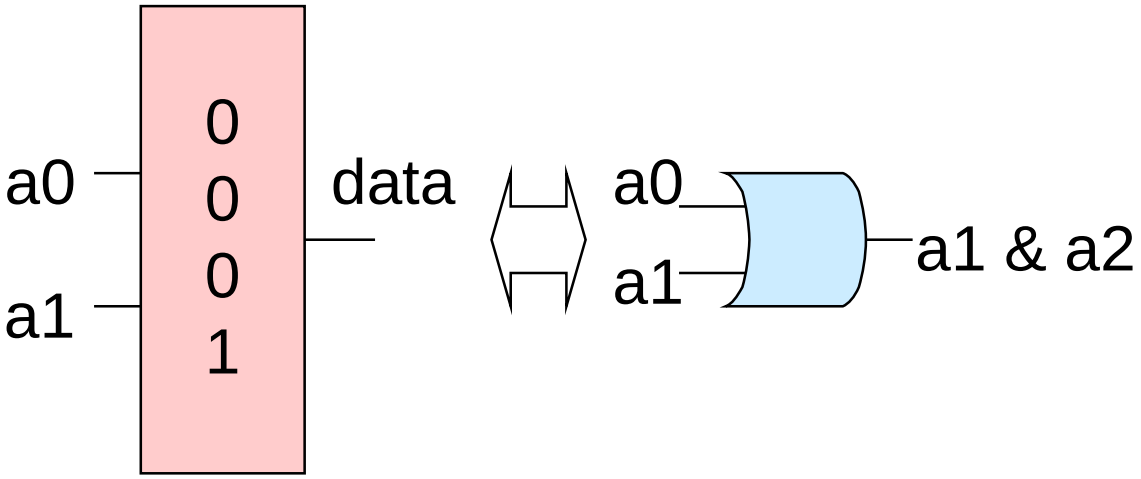
Source: In-stat April 1998

*Does not include software, hardwire or support EPROMs

What Is “Reconfigurable Hardware?”

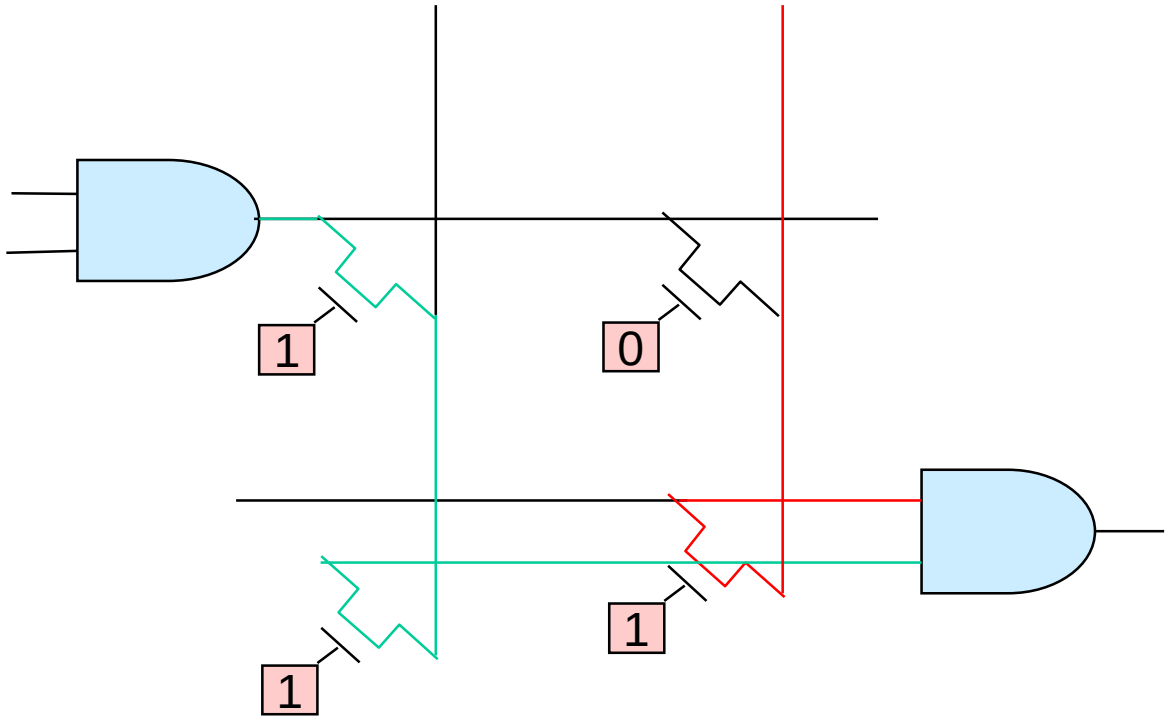


Basic Ingredient: RAM cell



Universal gate = RAM

Basic Ingredients (ctd)



A switch is controlled by a 1-bit RAM cell

Outline

- What is reconfigurable hardware
- RH vs other computation paradigms
- Challenges in RH research
- PipeRench: the CMU project:
  - the hardware
  - the software
- Conclusions

RH vs ASICs

- Generally Application-Specific Integrated Circuits will be faster than RH:
  - RH wires are slow & big
  - RH bit-slices are costly to interconnect
  - RH devices must store configuration on the chip

but

- RH can be reprogrammed
  - new algorithms
  - to fix bugs
- RH cheaper in small production
- RH tolerates faults better
- RH sometimes faster with staged computation

RH vs Microprocessors

- RH less flexible (like a VLIW with fixed instructions)

but

- RH provides more (customized) computation elements
- RH can decrease memory traffic
- RH can be tailored for specific algorithms and data types

RH will not replace mP, but complement them

Types of RH

- FPGAs: bit-level logic functionality

  (the basic processing elements compute on 1 bit)

- word-based architectures: PipeRench (CMU)

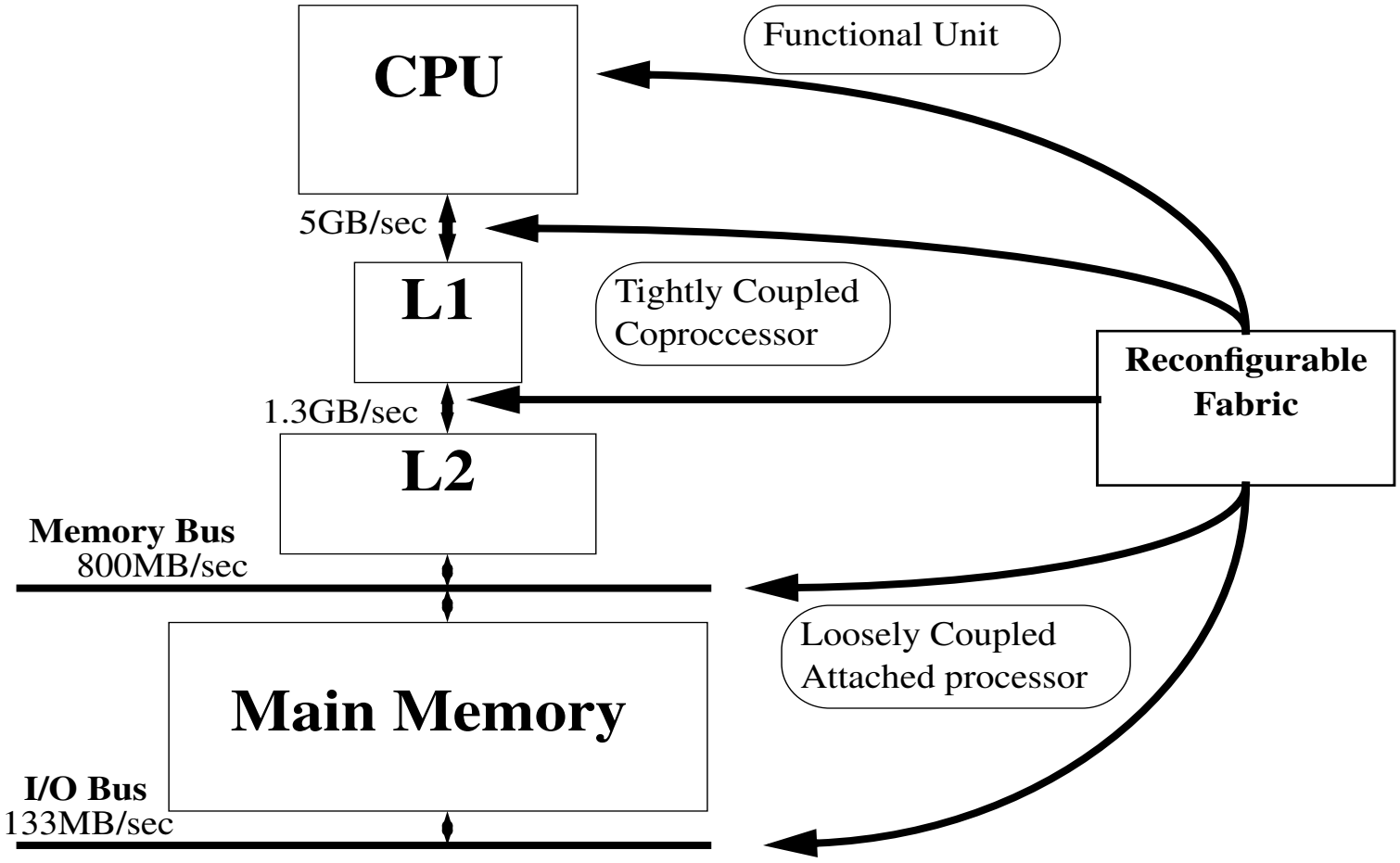
  (basic PE operates on 8 bits)

  (basic PE is a small ALU)

- coarse architectures: RAW (MIT)

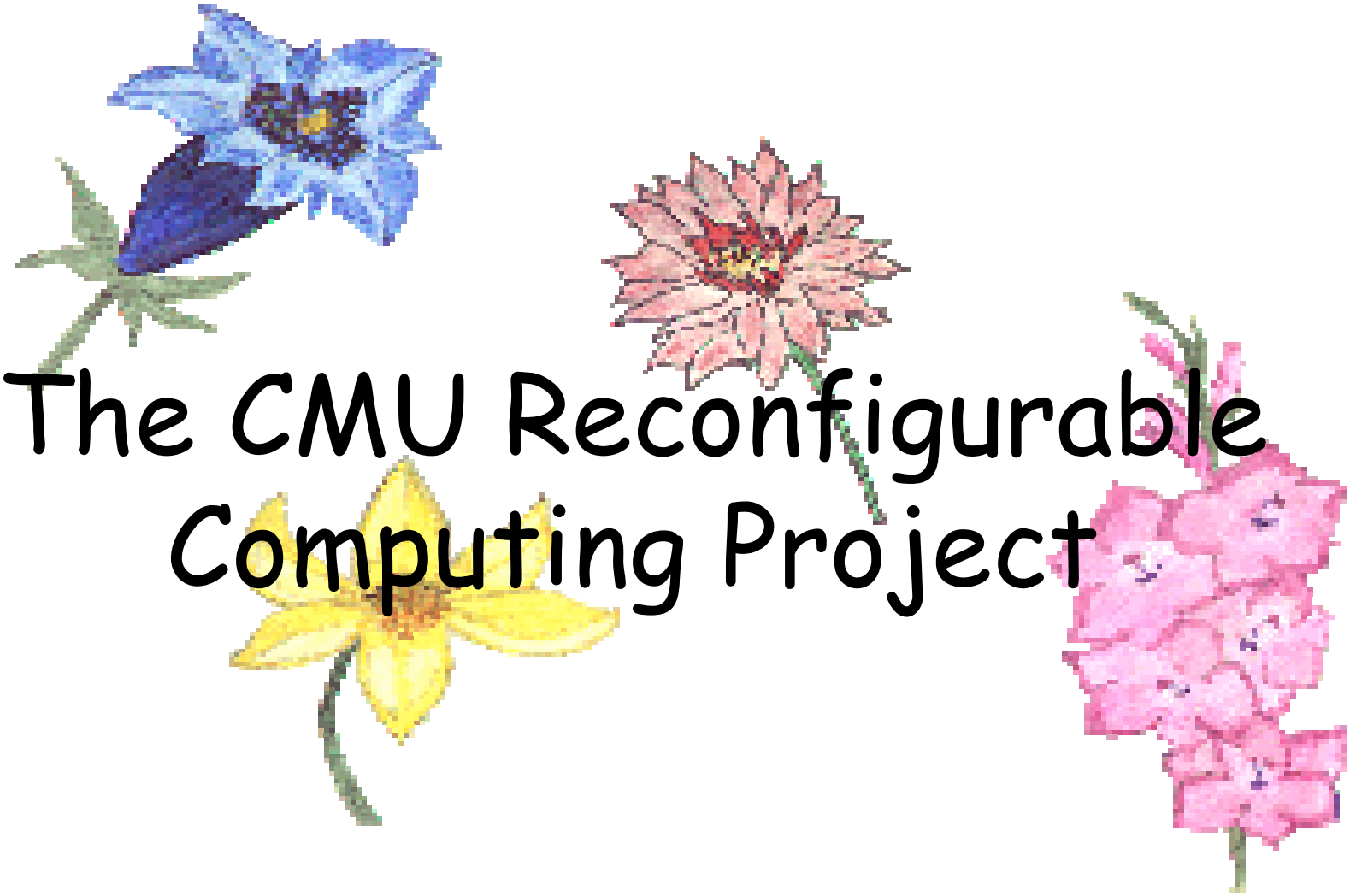
  (basic PE is a MIPS 2000 core)

RH In A System



Challenges In RC

- Software tools:
  - Programming RC like software development
  - Automatic compilation from HLL
  - Automatic program partitioning
- Mapping efficiently algorithms (no ISA)
- System issues
  - interfaces
  - find “ideal” RC fabric



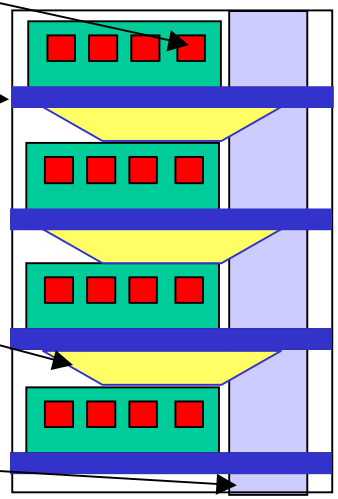
The CMU Reconfigurable Computing Project

Hardware Goals

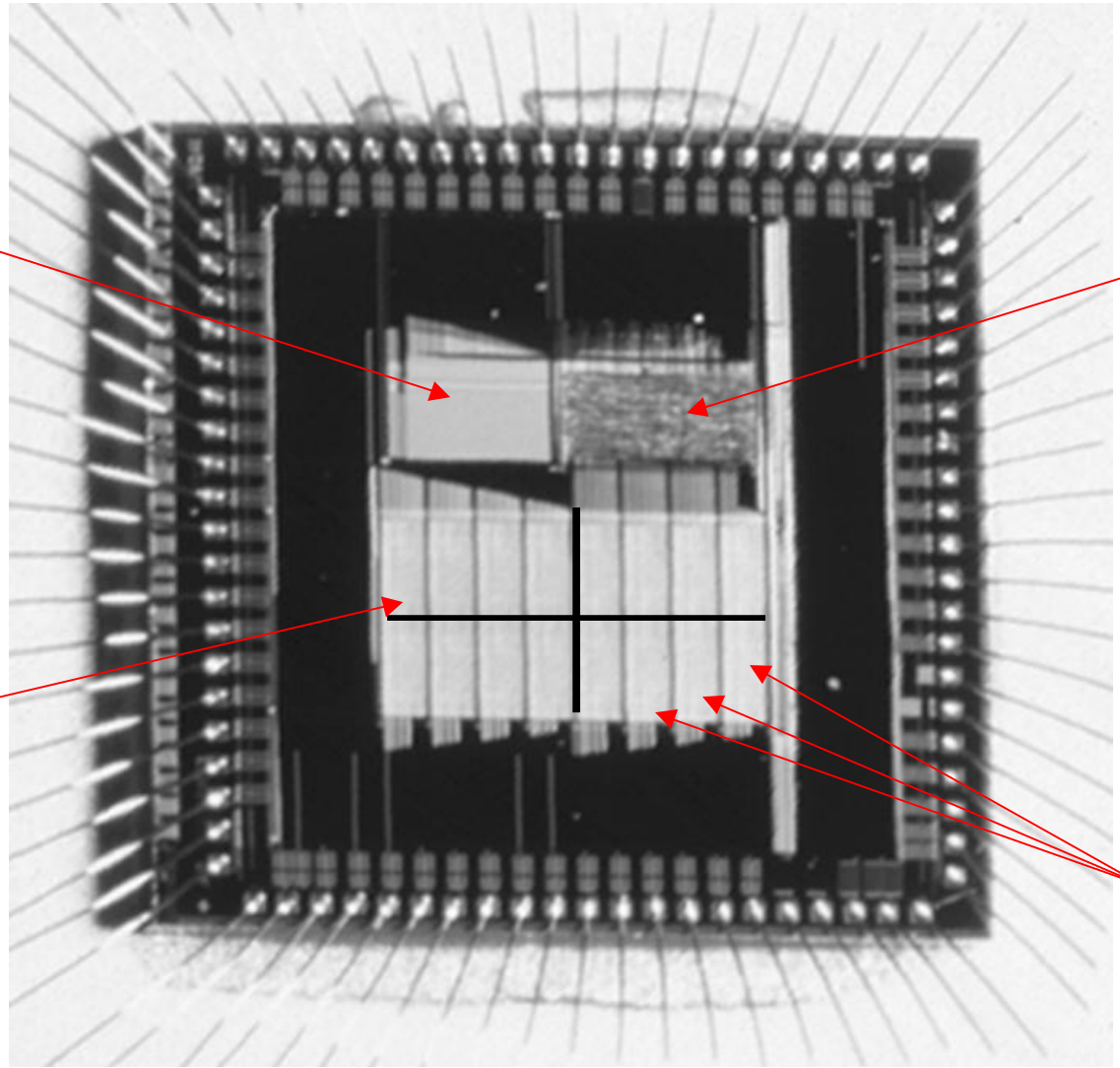
- To build a complete reconfigurable hardware device
- To build the system integration hardware
- To host the device in a PC

Our Device: PipeRench

- Word processing elements
- Pipelined architecture
- Virtualized hardware
- Local interconnection network
- Wide pipelined bus

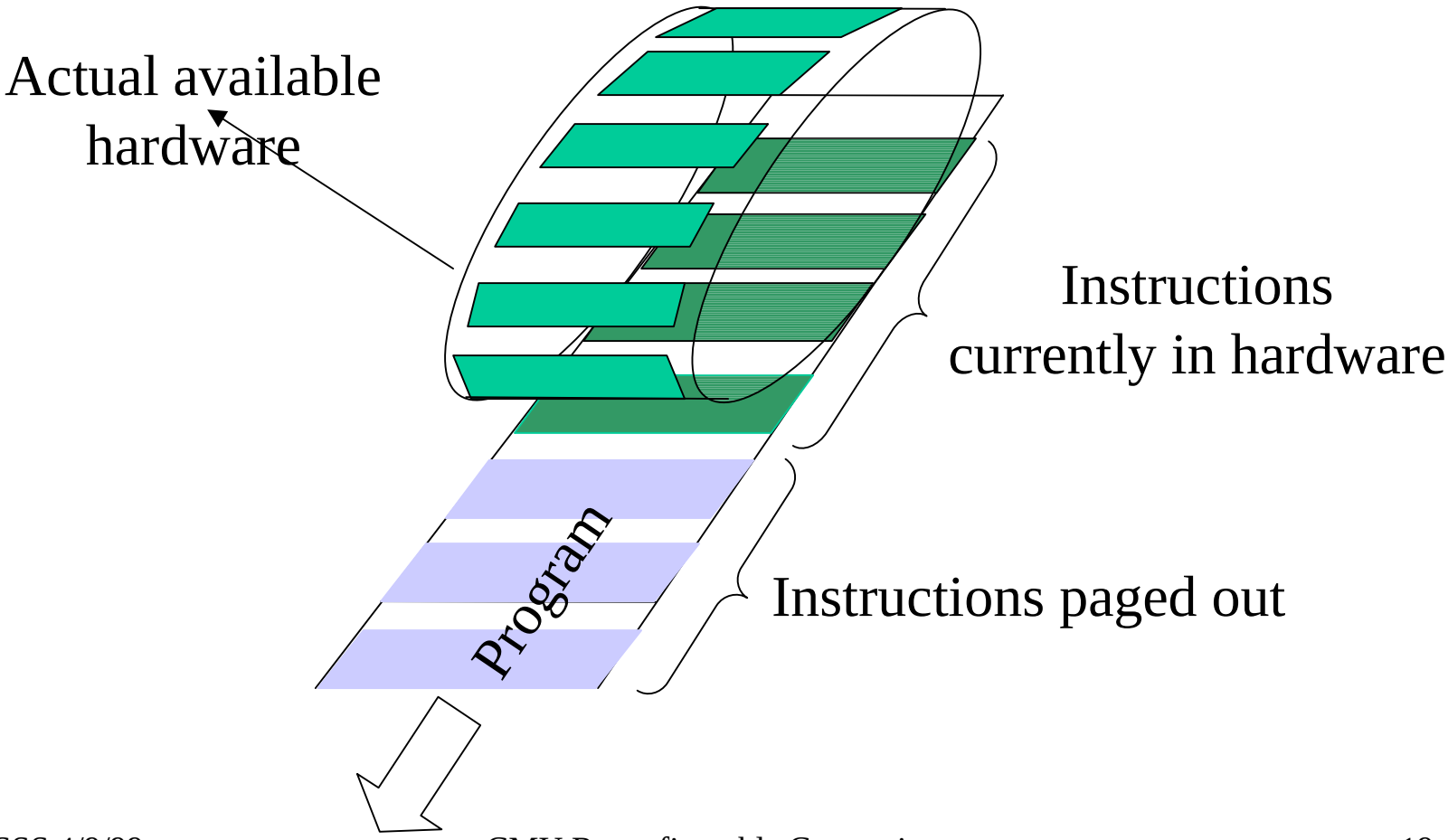


PipeRench

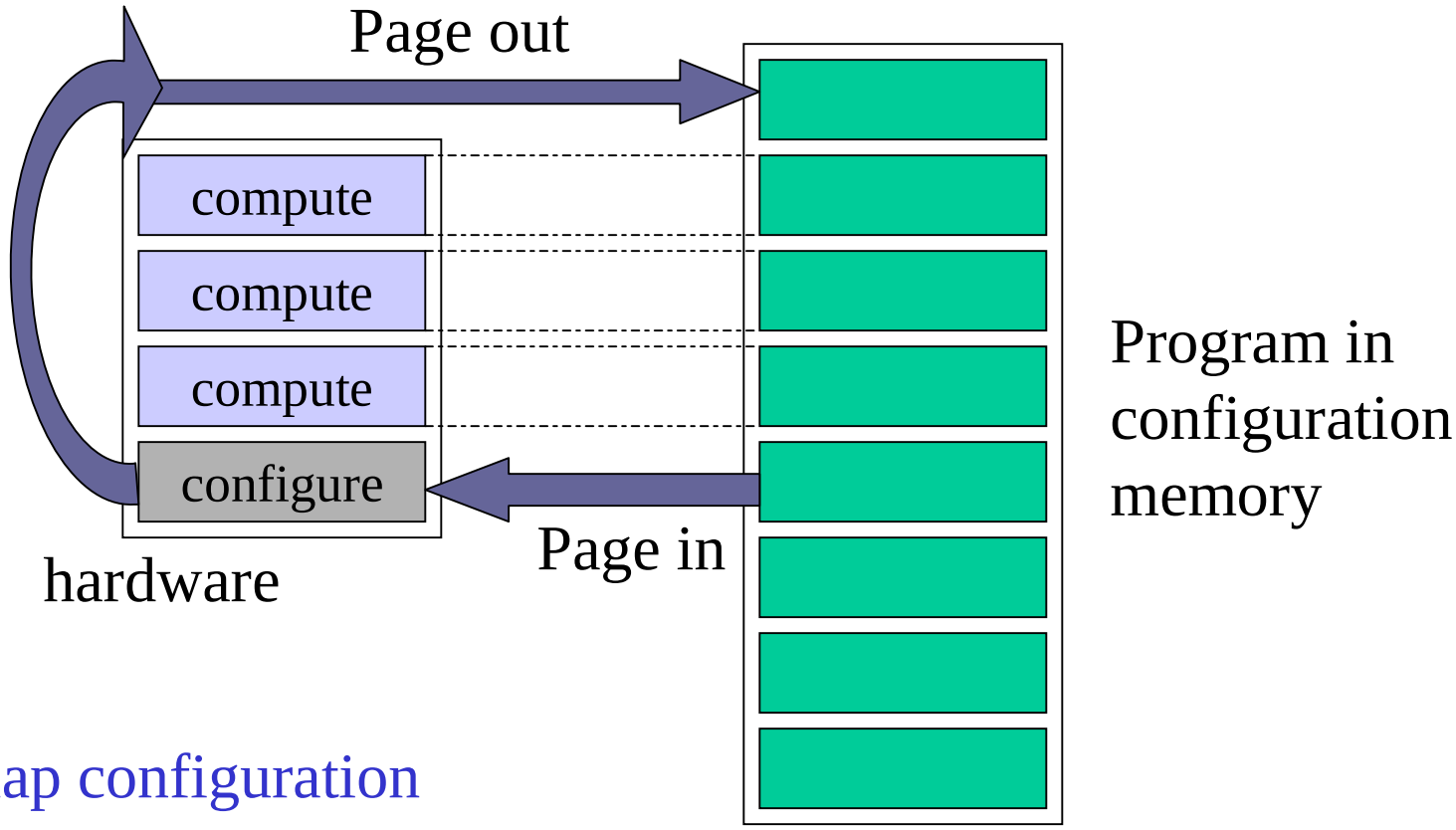

Configuration memory
Data & Config controller
Stripes
Processing elements

PipeRench

Hardware Virtualization

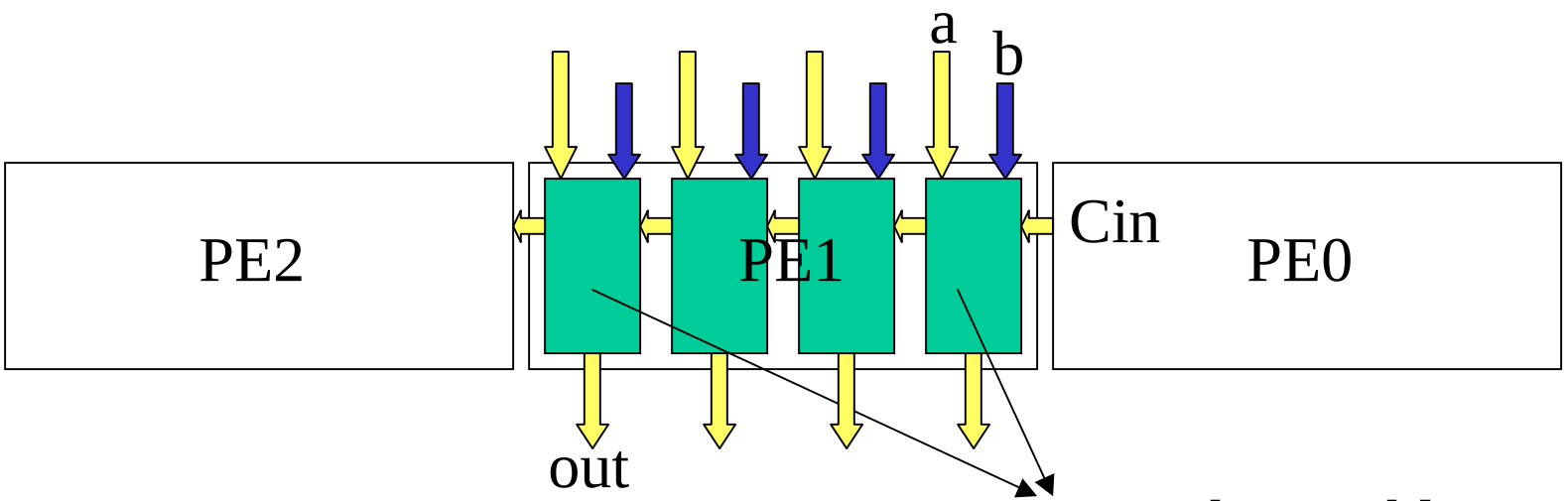


Hardware Virtualization (2)



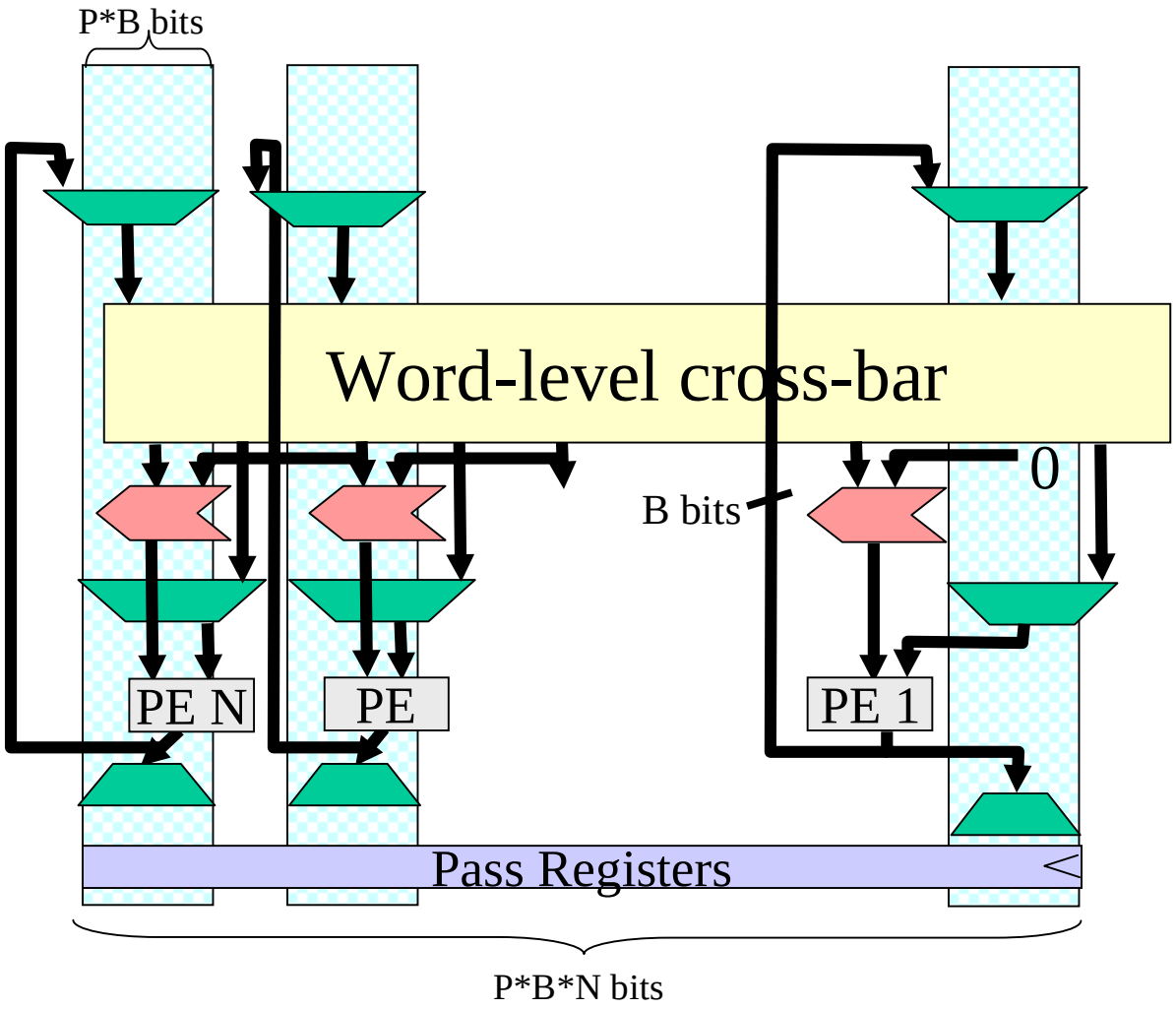
Overlap configuration with computation.

PipeRench Processing Elements

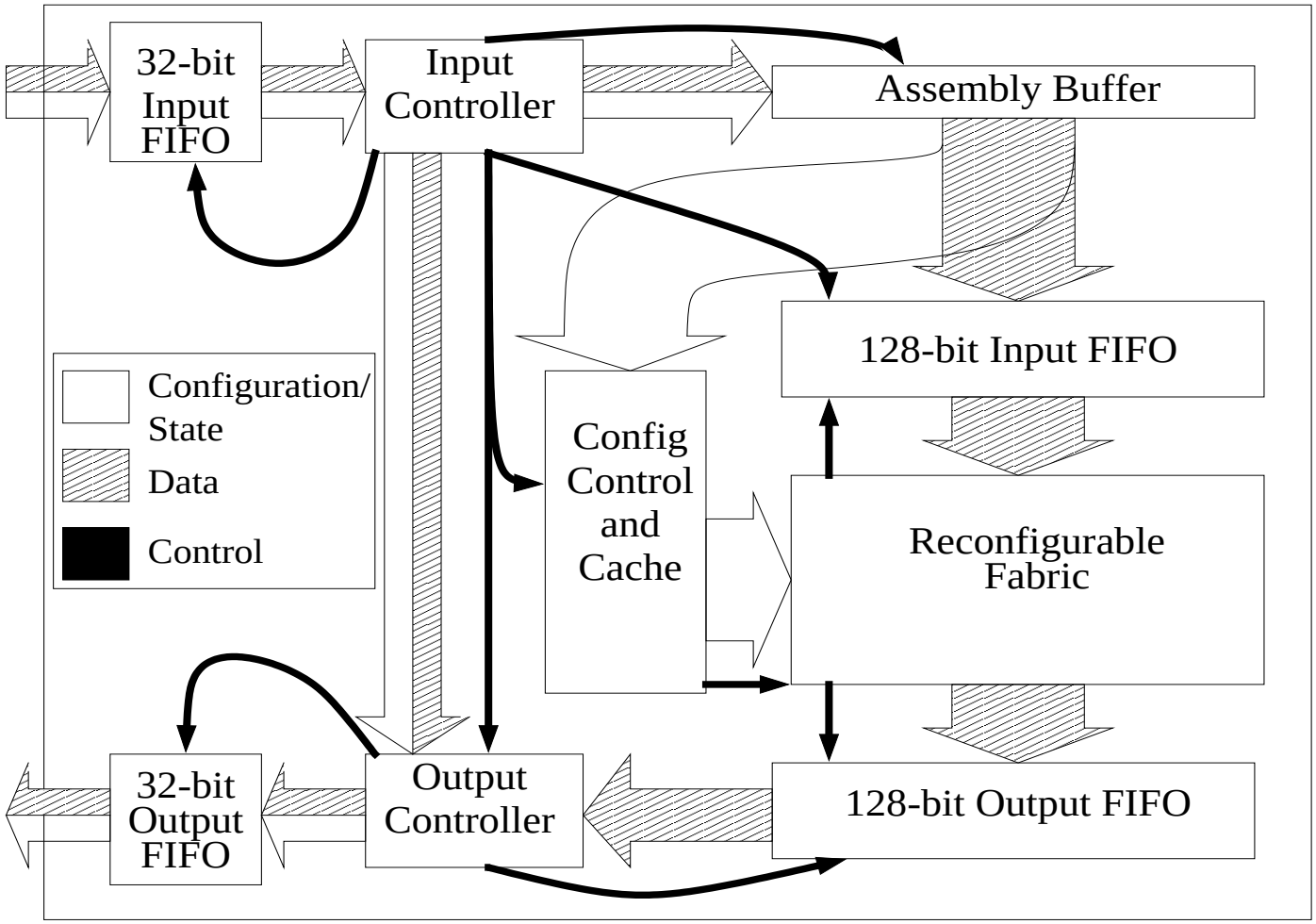


- Look-up table
- Any 3-to-1 function

The Interconnection Network

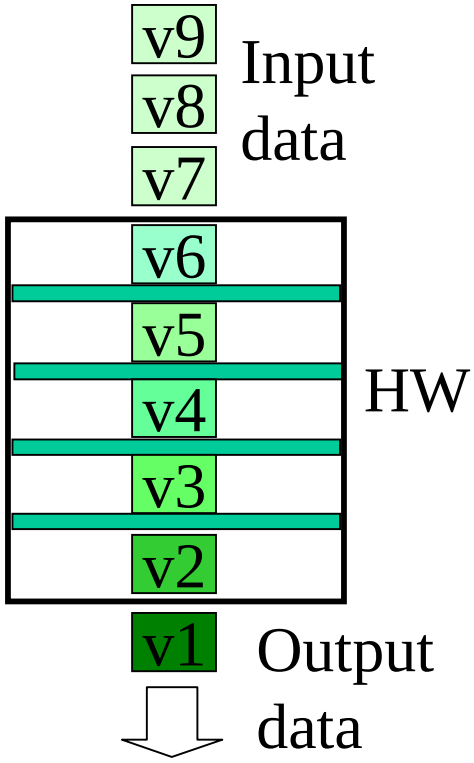


The PCI Board



Our Target Applications

- Pipelineable applications
  - Stream processing (e.g. DSP, encryption)
  - Multimedia processing
  - Vector processing
  - Limited data dependencies

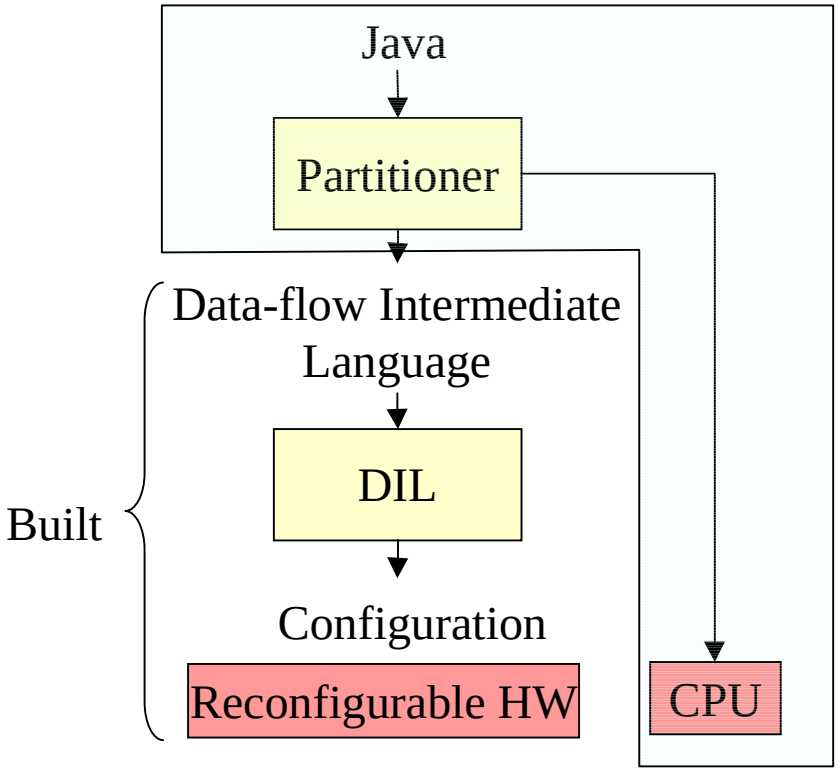


Computational power stems from massive parallelism

Software Goal

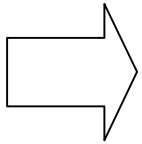
To program reconfigurable devices using the standard software development processes:

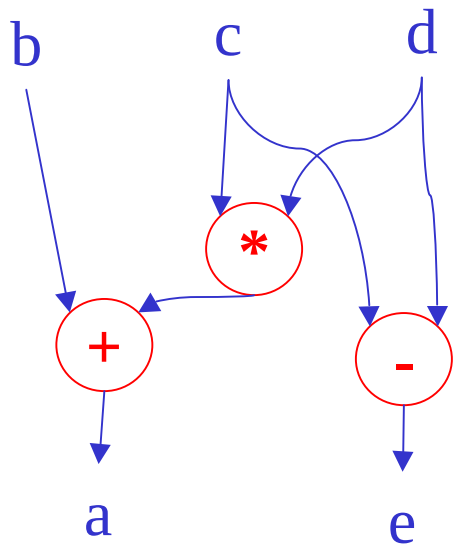
- Compile C or Java
- Do it quickly



Building Circuits From DIL

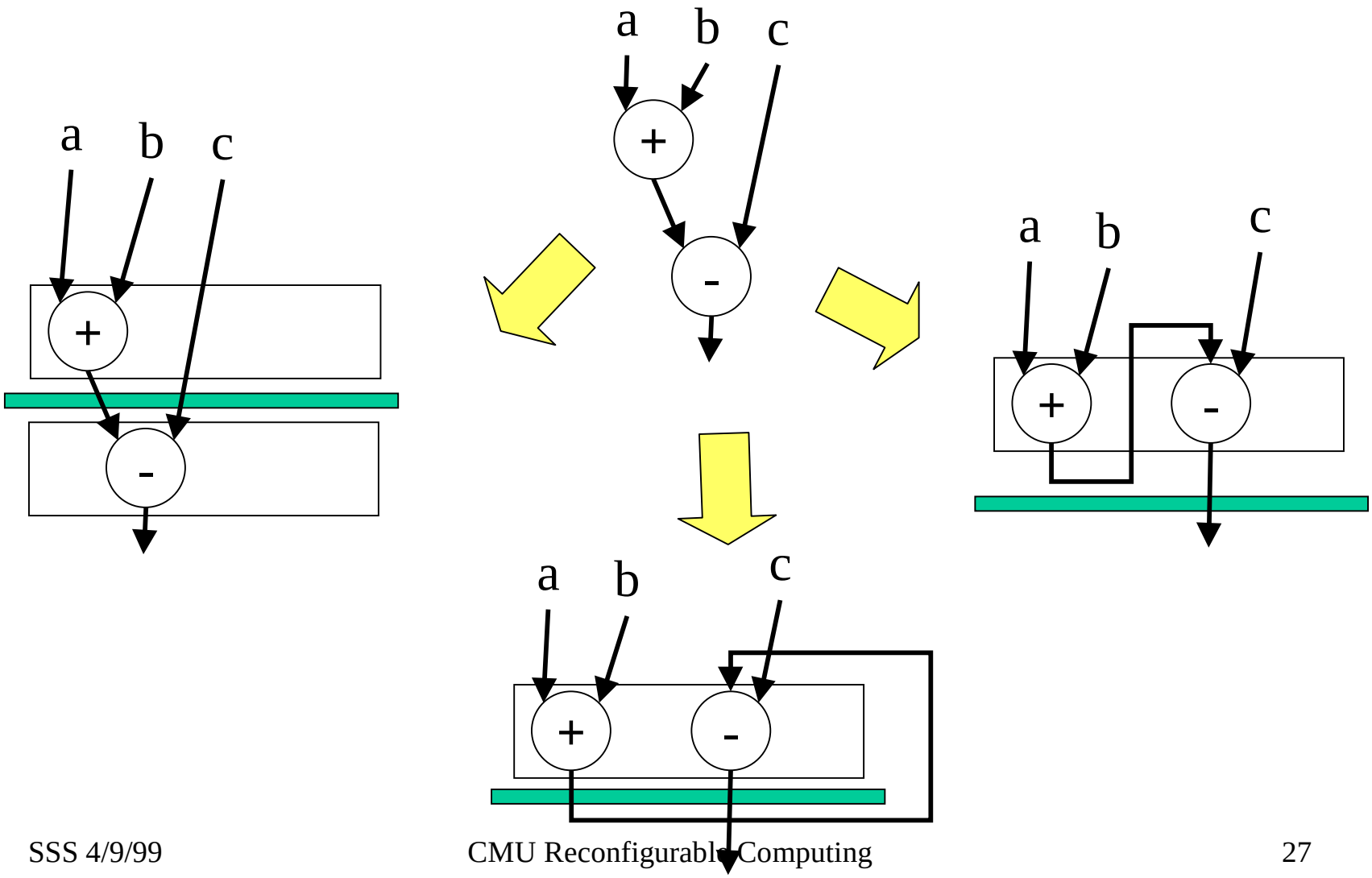
```
a = b + c * d;
e = c - d;
```



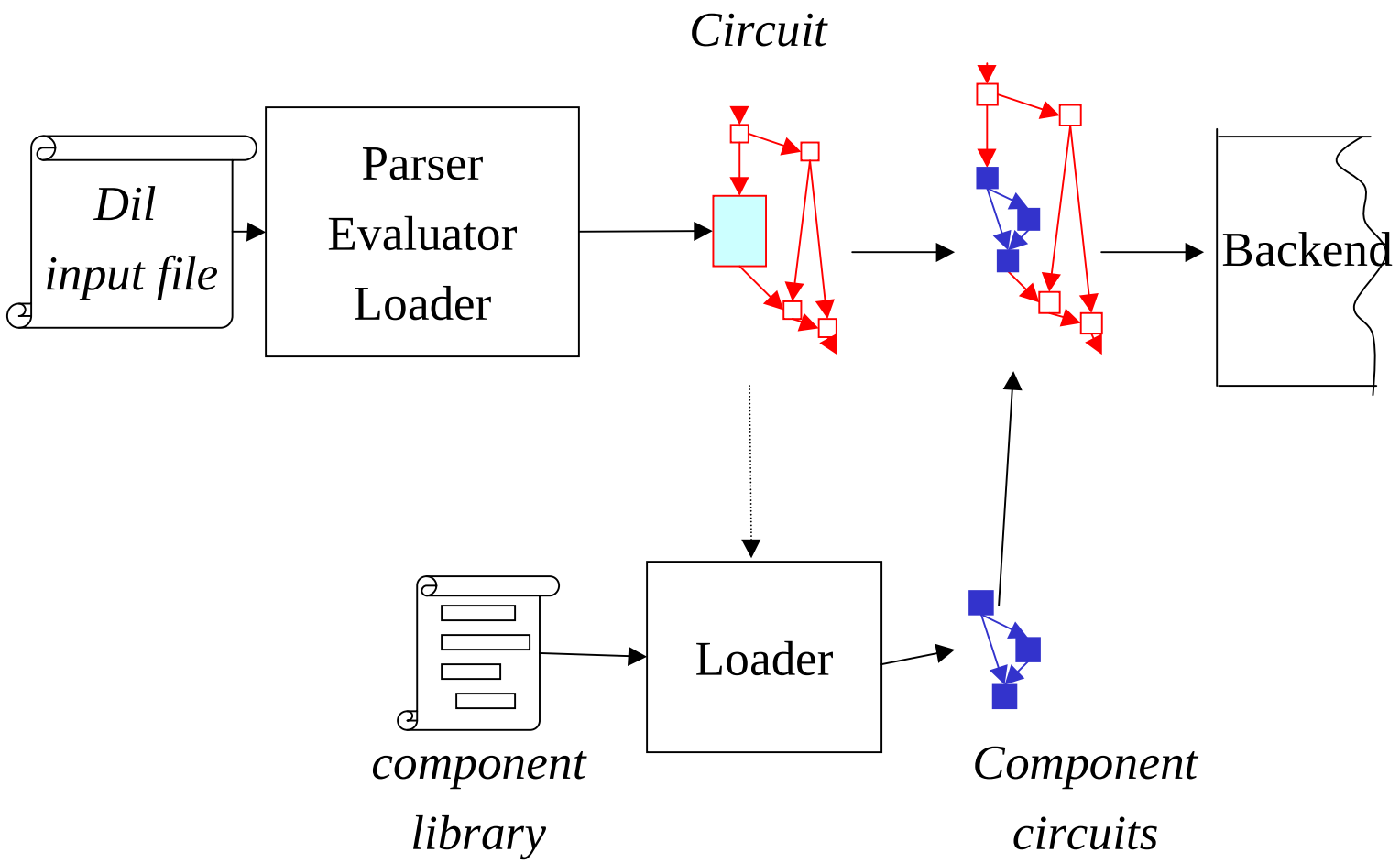


- variables ⇨ wires
- operators ⇨ gates

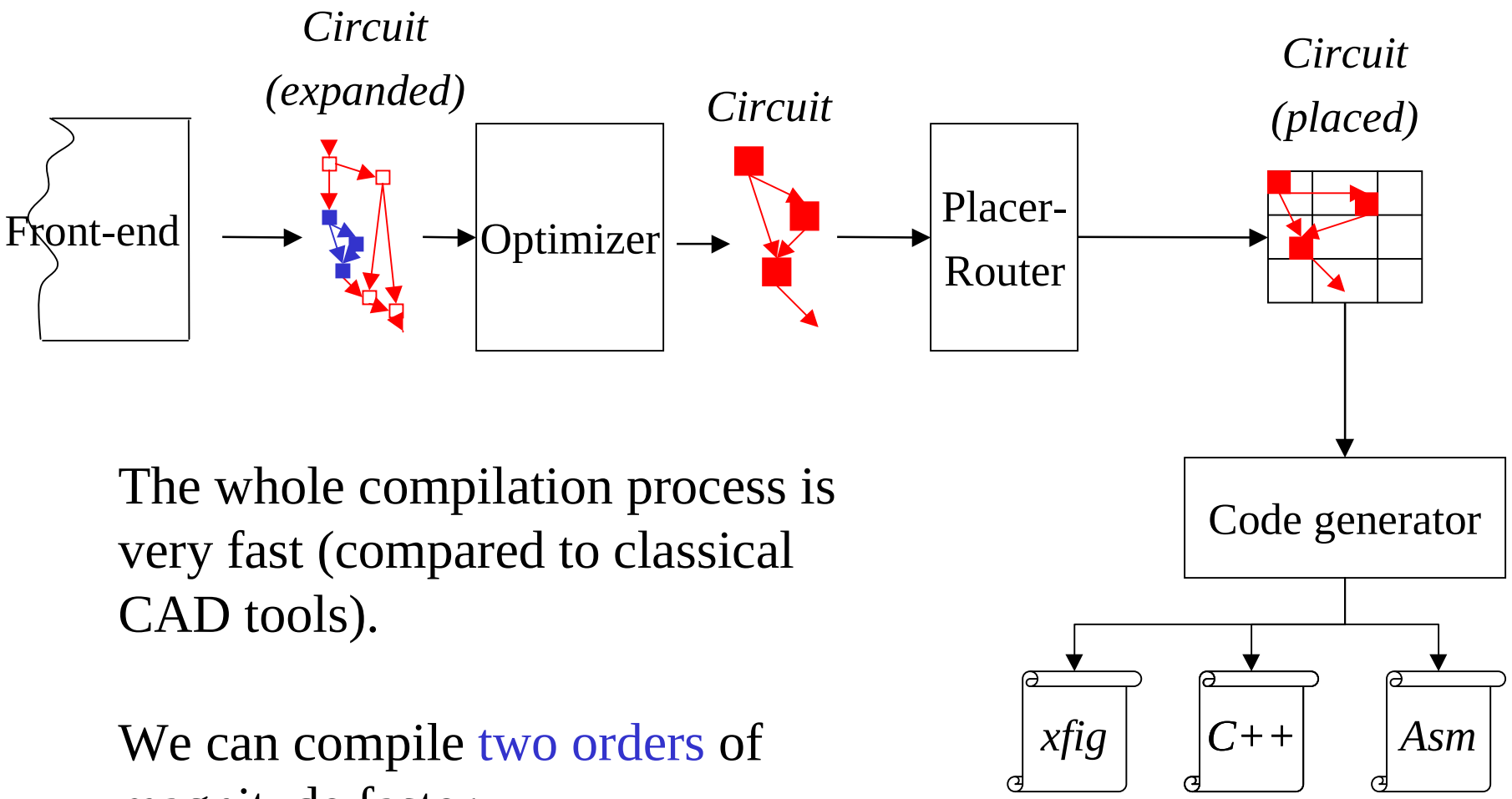
Mapping Circuits To PipeRench



The DIL Compiler Front-End



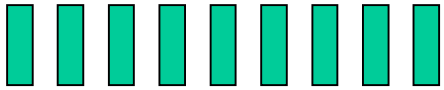
The DIL Compiler Backend

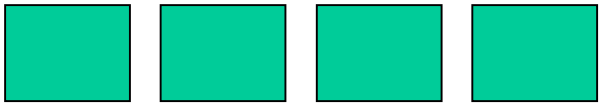


The whole compilation process is very fast (compared to classical CAD tools).

We can compile two orders of magnitude faster.

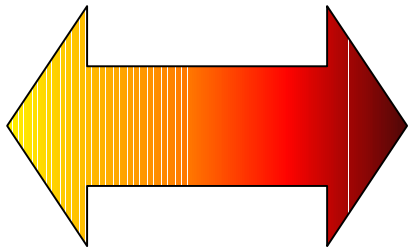
Processing Element Size Tradeoffs

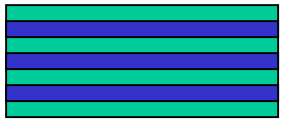


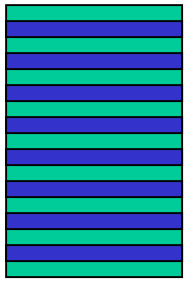


Small	Big
Efficient usage	Wasteful
Slower	Faster bit-slice
Flexible interconnect	Coarse routing
Bigger configuration	Fewer configuration bits
Place and route easier	Constrains the compiler

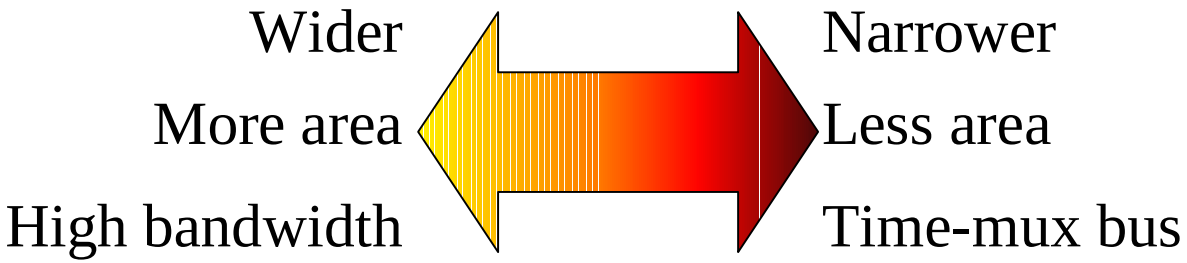
Stripe Width Tradeoffs

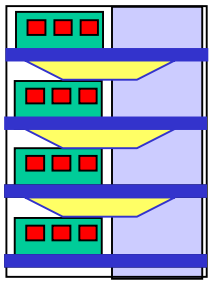
Wider		Narrower
Fewer stripes		More will fit
Virtualize more		Fewer page-ins
Bandwidth waste		Less bandwidth available
Placer freedom		Placement constrained

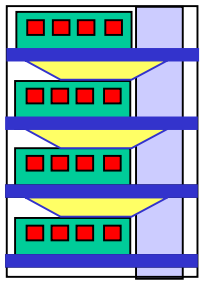




Bus Width Tradeoffs



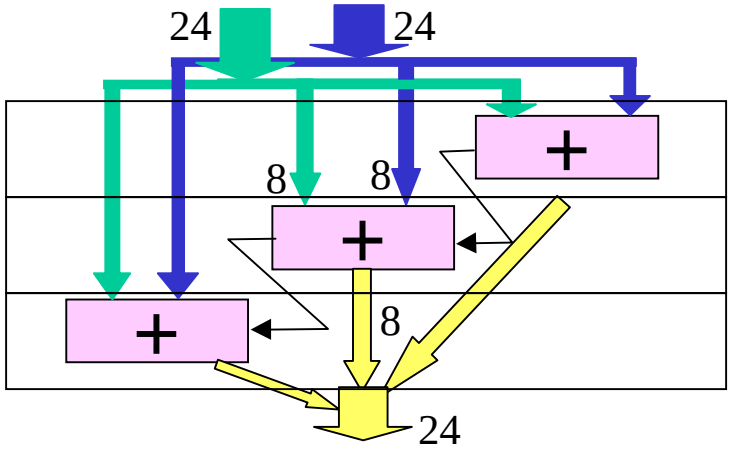


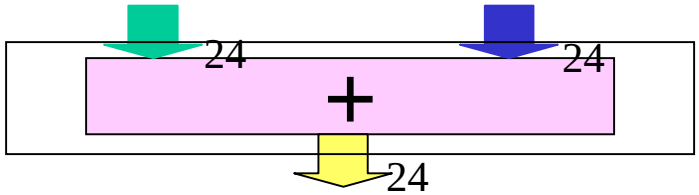


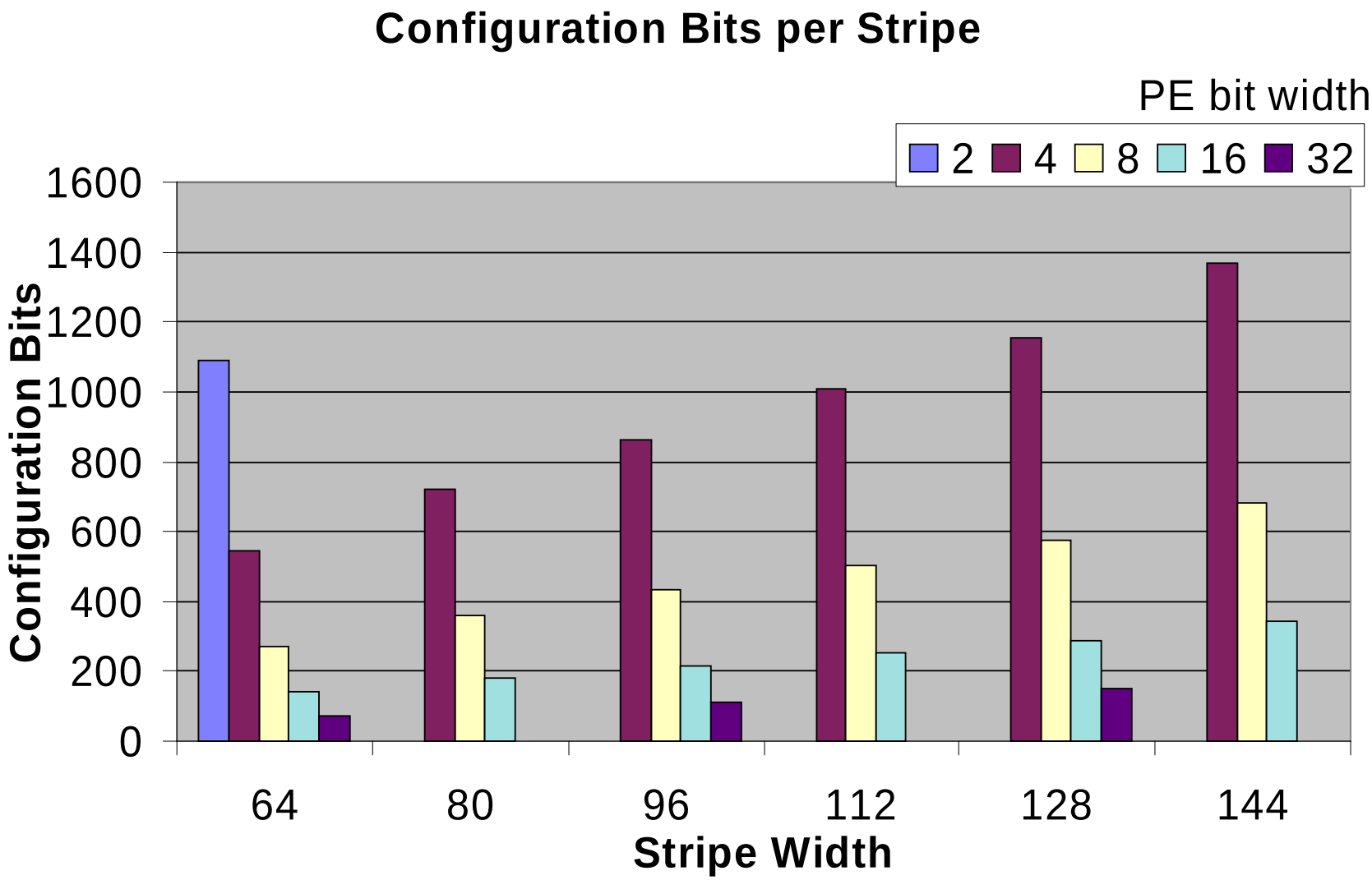
Clock Speed Tradeoffs

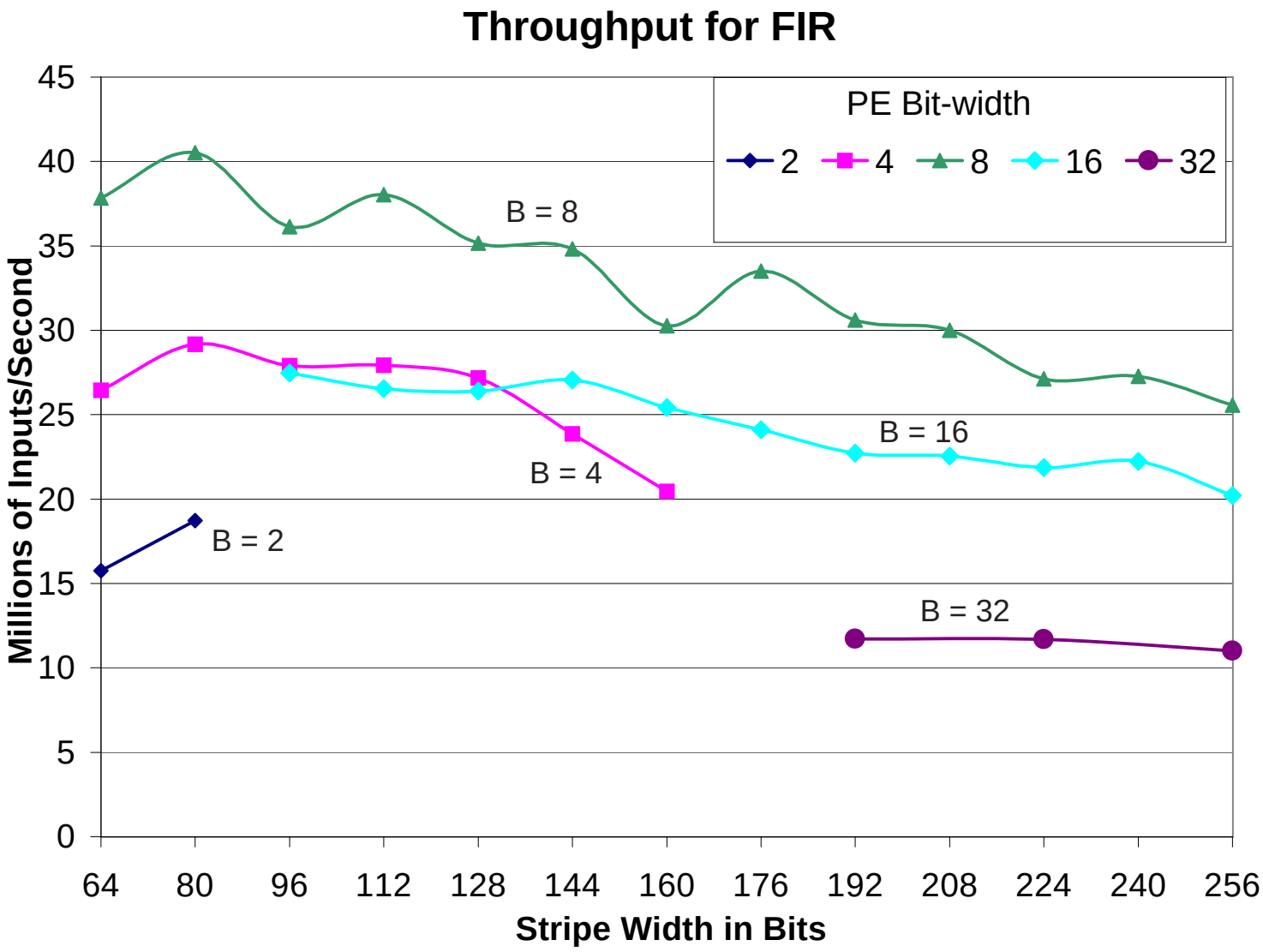
(run-time)

Faster		Slower
Short critical path		Big chains
Long pipeline built		Compact circuits
Decomposition overhead		Little decomposition
Virtualized more		Less virtualized





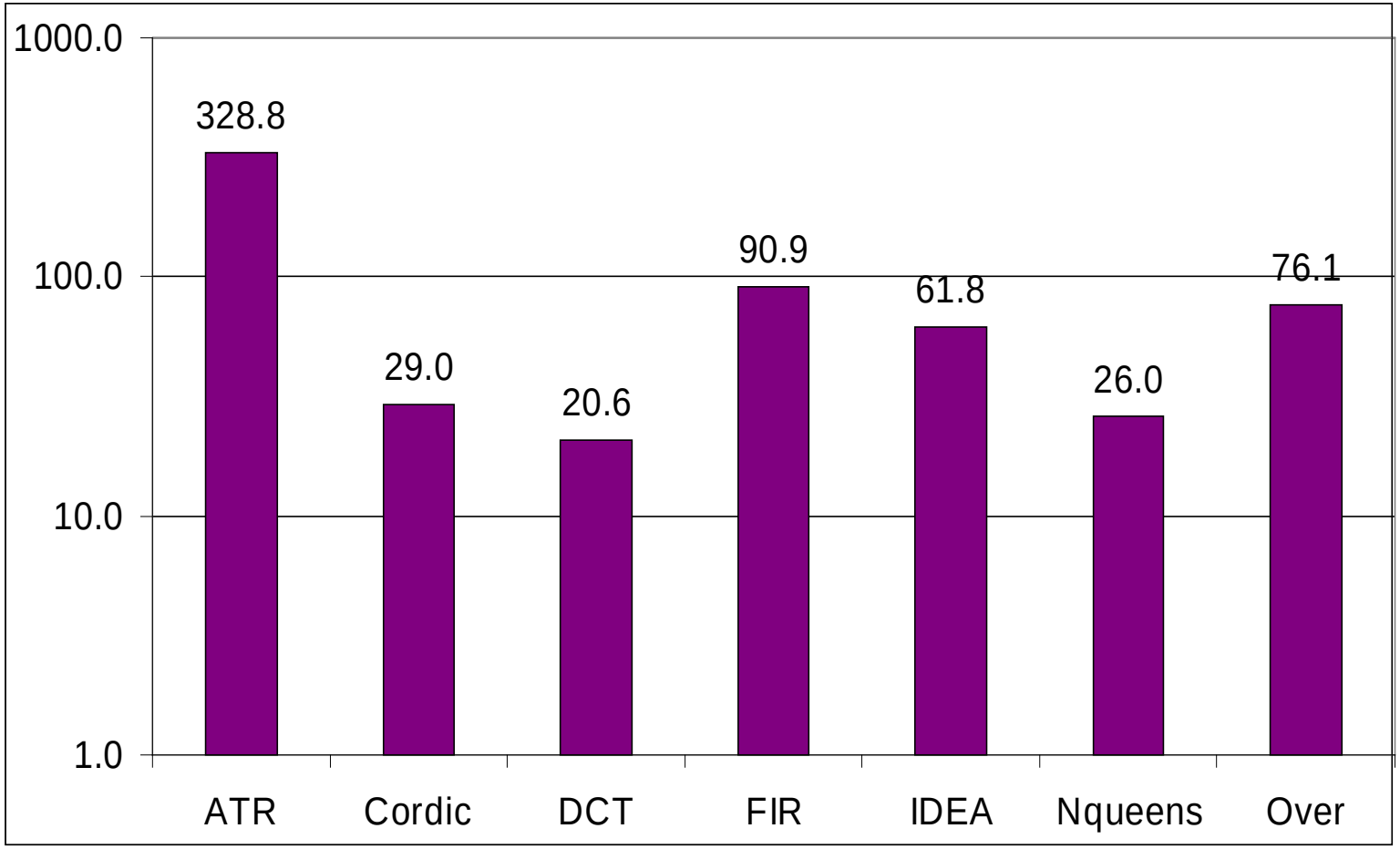

Configuration Bits per Stripe
PE bit width
2
4
8
16
32
1600
1400
1200
1000
800
600
400
200
0
Configuration Bits
64
80
96
112
128
144
Stripe Width


Throughput for FIR
PE Bit-width
2
4
8
16
32
B = 8
B = 4
B = 2
B = 16
B = 32
Millions of Inputs/Second
0
5
10
15
20
25
30
35
40
45
64
80
96
112
128
144
160
176
192
208
224
240
256
Stripe Width in Bits

Project Status

- Operational:
  - Behavioral and structural models of Piperench in Verilog
  - Assembler, simulator
  - Tools for visualization and debugging
  - One tile fabricated and tested
  - Very fast compiler from intermediate language
- In work:
  - Prototype PipeRench to be taped this summer
  - PCI board to host PipeRench in a PC

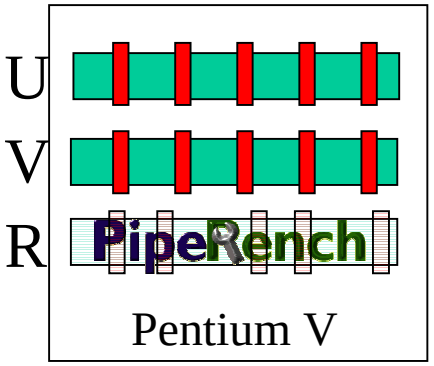
Simulated Speed-up vs. UltraSparc @ 300Mhz



Future Work

- Build the PCI board
- Build the OS device drivers
- Start investigating HLL issues:
  - automatic partitioning
  - translation to DIL
  - special code transformations

Conclusions

- A set of important applications can benefit from RC devices
- RC offer potential for substantial performance improvement at a low cost

- RC devices will soon be mainstream in the embedded computing world; perhaps in the future they will also permeate the desktop