

# CS184b: Computer Architecture (Abstractions and Optimizations)

Day 14: May 2, 2005  
Vector, SIMD



## Today

- Data Parallel
  - Model
  - Application
  - Resources
  - Architectures
    - Abacus
    - T0

## Data Parallel Model

- Perform same computation on multiple, distinct data items
  - Sequential set of operations (like ISA)
  - ...but on large aggregate collection
- SIMD
  - recall simplification of general array model
  - every PE get same instruction
    - feed large number of PEs with small instruction bandwidth

CS184a

## Architecture Instruction Taxonomy

Control Threads (PCs)		Instructions per Control Thread		Architecture/Examples
Instruction Depth	Granularity	Instruction Depth		
		0	0	n/a
0	1	w	1	FPGA
			$n_c \cdot 1$	Reconfigurable ALUs
			$n_c \cdot w$	Bitwise SIMD
1	c	w	$n_c \cdot 1$	Traditional Processors
			$n_c \cdot w$	Vector Processors
			1	DPGA
1	c	w	8	PADDI
			16	VLIW
			1	HSRA/SCORE
m	1	w	$n_c \cdot w$	MSIMD
			1	VEGA
			8	PADDI-2
m	1	w	16	MIMD (traditional)
			c	

## Example

- Operations on vectors
  - vector sum
  - dot, cross product
  - matrix operations
- Simulations / finite element / cellular automata
  - same update computation on every site
- Image/pixel processing
  - compute same thing on each pixel

## Model

- Zero, one, infinity
  - good model has unbounded number of processors (data parallel items)
  - user allocates virtual processors
  - folded (as needed) to share physical processors

## Vector Model Success

- Gave programmers a **simple** model of the machine
  - It can do operations on vectors quickly
- Programmers just think about expressing as vector ops
- Compilers...
  - Arguably never that good as creating vectors

Caltech CS184 Spring2005 -- DeHon

7

## How do an `if`?

- Have large set of data
- How do we conditionally deal with data?

Caltech CS184 Spring2005 -- DeHon

8

## Branchless Multiply Example

- Recall writing multiplication without branching
  - ...like hardware/spatial
  - ...need to mask and multiplex
  - Local Control

Caltech CS184 Spring2005 -- DeHon

9

## Key: Local State

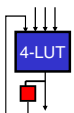
- Set state during computation
- Use state to modify transmitted instruction
  - Operation should simply be  $PE.op(inputs, state)$
  - Often mask
    - select subset of processors to operate
    - like predicated operations in conventional processor

Caltech CS184 Spring2005 -- DeHon

10

## Local State Op

- Consider 4-LUT with two states
  - w/ local state bit, can implement a 3-LUT function with one state bit
  - state bit is 4th input to LUT can decide which operation to perform



Caltech CS184 Spring2005 -- DeHon

11

## ABS with Mask

- $tmp = val < 0$
- $rval = val$
- mask all processors with  $tmp == true$
- $rval = -(val)$
- unmask

Caltech CS184 Spring2005 -- DeHon

12

## Model

- Model remains
  - all PEs get same operation
  - compute on local state with operation

Caltech CS184 Spring2005 -- DeHon

13

## Synchronization

- Strong SIMD model
  - all operations move forward in lock-step
  - don't get asynchronous advance
  - don't have to do explicit synchronization

Caltech CS184 Spring2005 -- DeHon

14

## Communications

- Question about how general
- Common, low-level
  - nearest-neighbor
  - cheap, fast
  - depends on layout...
  - effect on virtual processors and placement?

Caltech CS184 Spring2005 -- DeHon

15

## Communications

- General network
  - allow model with more powerful shuffling
  - how rich? (expensive)
  - wait for longest operation to complete?
- Use Memory System?

Caltech CS184 Spring2005 -- DeHon

16

## Memory Model?

- PEs have local memory
- Allow PEs global pointers?
- Allow PEs to dereference arbitrary addresses?
  - General communications
  - Including conflicts on PE/bank
    - potentially bigger performance impact in lock-step operation
- Data placement important

Caltech CS184 Spring2005 -- DeHon

17

## Vector Model

- Vector is primary data structure
- Memory access very predictable
  - easy to get high performance on
    - e.g. burst memory fetch, banking
  - one address and get stream of data

Caltech CS184 Spring2005 -- DeHon

18

## ...not always that simple...

- Often trick to making vector model apply to problems is rich data access
  - Need interconnect to permute data below the vector level
  - Typically:
    - **Gather:** create vector from this set of addresses....
    - **Scatter:** write the vector out to this set of addresses

Caltech CS184 Spring2005 -- DeHon

19

## How effect control flow? (SIMD and Vector)

- Predicated operations take care of local flow control variations
- Sometimes need to effect entire control stream
- E.g. relaxation convergence
  - compute updates to refine some computation
  - until achieve tolerance

Caltech CS184 Spring2005 -- DeHon

20

## Flow Control

- Ultimately need one bit (some digested value) back at central controller to branch upon
- How get?
  - Pick some value calculated in memory?
  - Produce single, aggregate result

Caltech CS184 Spring2005 -- DeHon

21

## Reduction Value

- Example: summing-or
  - Or together some bit from all PEs
    - build reduction tree....log depth
  - Typical usage:
    - processor asserts bit when find solution
    - processor deassert bit when solution quality good enough
      - detect when all processors done

Caltech CS184 Spring2005 -- DeHon

22

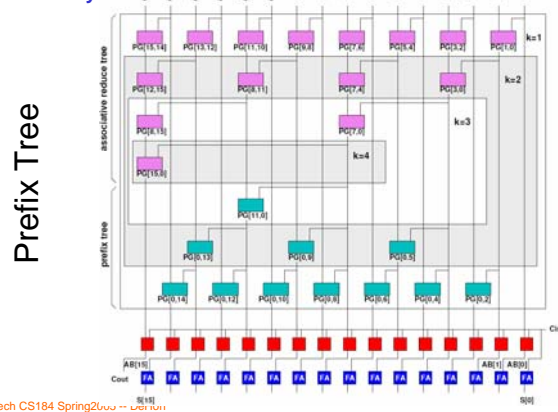
## Key Algorithm: Parallel Prefix

- Often will want to calculate some final value on aggregate
  - E.g. dot product: sum of all pairwise products
  - Already saw in producing
    - log-depth carries
    - Arbitrary LUT cascades
    - Ultrascalar register updates

Caltech CS184 Spring2005 -- DeHon

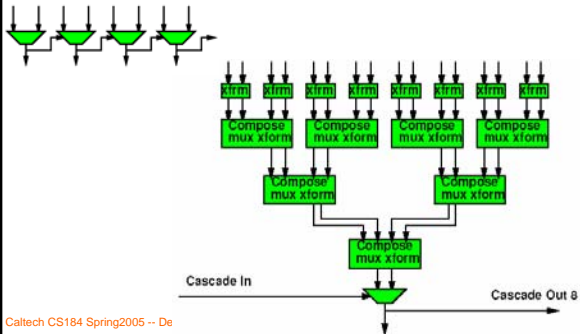
23

## CS184a: Day 3



Caltech CS184 Spring2005 -- DeHon

## Parallel Prefix Mux-cascade



## Parallel Prefix

- Calculate **all** intermediate results in log depth
  - e.g. all intermediate carries
  - e.g. all sums to given point in vector
- More general than tree reduction
  - tree reduction (sum, or, and) uses commutativity
  - parallel prefix only requires associativity

## Parallel Prefix...

- Count instances with some property
  - Locally identify property
  - Then do prefix sum
- Parsing
- List operations
  - pointer jumping, find length, matching

## Terms

- SIMD – Single Instruction Multiple Data
- Vector – SIMD on 1D array of words
- SPMD – Single Program Multiple Data
  - Coined to name the programming model separate from the machine/execution model
  - (may use SPMD model on MIMD machine)

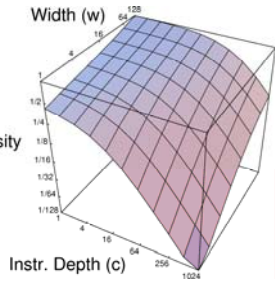
## Resources

## Contrast VLIW/SS

- Single instruction shared across several ALUs
  - (across more bits)
- Significantly lower control
- Simple/predictable control flow
- Parallelism (of data) in model

## Peak Densities from Model

- Only 2 of 4 parameters
  - small slice of space
  - 100x density across
- Large difference in peak densities
  - large design space!



31

## Calibrate Model

<b>FPGA</b>	<b>model</b> $w = 1, d = c = 1, k = 4$	<b>880K</b> $\lambda^2$
	<b>Xilinx 4K</b>	<b>630K</b> $\lambda^2$
	<b>Altera 8K</b>	<b>930K</b> $\lambda^2$
<b>SIMD</b>	<b>model</b> $w = 1000, c = 0, d = 64, k = 3$	<b>170K</b> $\lambda^2$
	<b>Abacus</b>	<b>190K</b> $\lambda^2$
<b>Processor</b>	<b>model</b> $w = 32, d = 32, c = 1024, k = 2$	<b>2.6M</b> $\lambda^2$
	<b>MIPS-X</b>	<b>2.1M</b> $\lambda^2$ 32

## Examples

33

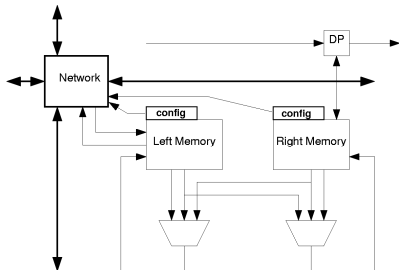
## Abacus: bit-wise SIMD

- Collection of simple, bit-processing units
- PE:
  - 2x3-LUT (think adder bit)
  - 64 memory bits, 8 control config
  - active (mask) register
- Network: nearest neighbor with bypass
- Configurable word-size

[Bolotski et al. ARVLSI'95]

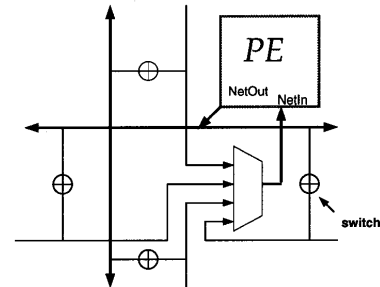
34

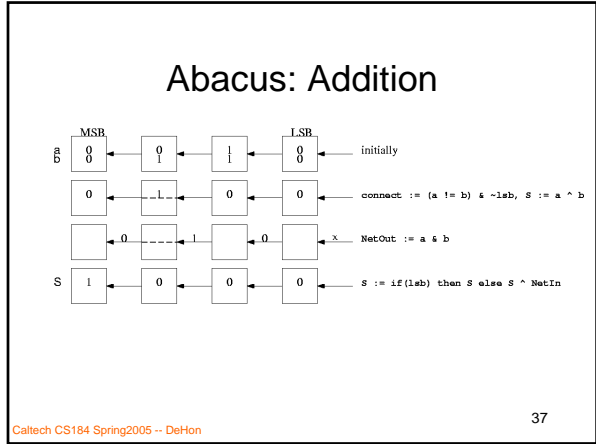
## Abacus: PE



35

## Abacus: Network





### Abacus: Scan Ops

$A = [4 \ 1 \ 7 \ 8 \ 3 \ 2 \ 1 \ 5]$   
 $+scan = [4 \ 5 \ 12 \ 20 \ 23 \ 25 \ 26 \ 31]$   
 $maxscan = [4 \ 4 \ 7 \ 8 \ 8 \ 8 \ 8 \ 8]$   
 $+reduce = 31$

Caltech CS184 Spring2005 -- DeHon

- ### Abacus: bit-wise SIMD
- High raw density:
    - 660 ALU Bit Ops/ $\lambda^2$ s
    - Compare peak of 10 bops/ $\lambda^2$ s for proc.
    - Compare ~100 bops/ $\lambda^2$ s for FPGAs
  - Do have to synthesize many things out of several operations
  - Nearest neighbor communication only
- Caltech CS184 Spring2005 -- DeHon

### Abacus: Cycles

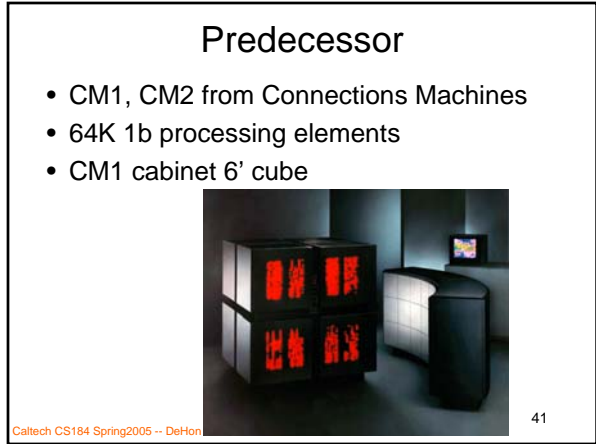
Operation	8-bit		16-bit		32-bit	
	Cycles	GOPS	Cycles	GOPS	Cycles	GOPS
Add	4	4.0	4	2.0	5	0.7
Shift	2	8.0	2	4.0	2	2.0
Accumulate	3	5.2	3	2.6	3	1.3
Move	3	5.2	4	2.0	6	0.6
Compare	6	2.6	11	0.6	12	0.2
Multiply (16 × 16)					180	0.03

Algorithm	Cycles	Time ( $\mu$ sec)
Edge Detection $\sigma = 1.6$	380	3
Optical Flow, $\Delta = 2, 5 \times 5$ region	3000	24
Surface Reconstruction (1 iteration)	370	3

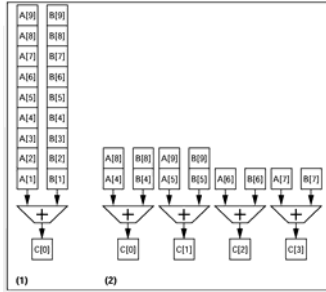
1  $\mu$ m CMOS (6.5mm x 7.3mm, 1000 PEs)

Caltech CS184 Spring2005 -- DeHon



- ### T0: Vector Microprocessor
- Word-oriented vector pipeline
  - Scalable vector abstraction
    - vector ISA
    - size of physical vector hardware abstracted
  - Communication mostly through memory
- [Asanovic *et al.*, IEEE Computer 1996]  
 [Asanovic *et al.*, Hot Chips 1996]
- Caltech CS184 Spring2005 -- DeHon

## Vector Scaling



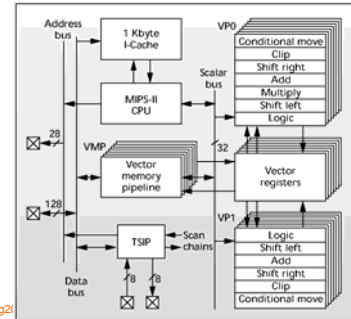
No element-to-element dependence:  
 • Avoid pipeline interlock  
 • Easy parallel dispatch

Just dependence  
 vector-op to vector-op

Caltech CS184 Spring2005 -- DeHon

43

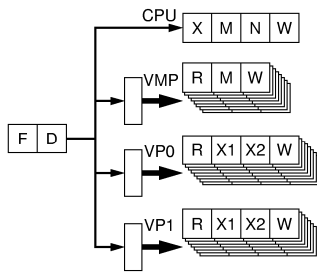
## T0 Microarchitecture



Caltech CS184 Spring2005

44

## T0 Pipeline



Caltech CS184 Spring2005 -- DeHon

45

## T0 ASM example

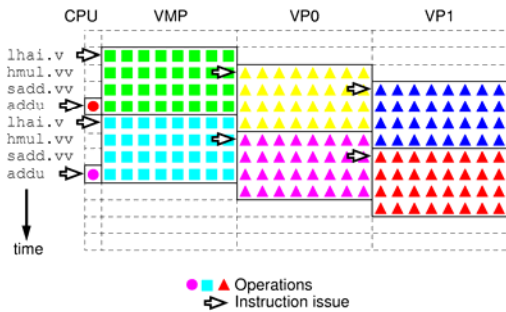
```

lhai.v vv1, t0, t1      # Vector load.
hmul.vv vv4, vv2, vv3  # Vector mul.
sadd.vv vv7, vv5, vv7  # Vector add.
addu t2, -1            # Scalar add.
lhai.v vv2, t0, t1      # Vector load.
hmul.vv vv5, vv1, vv3  # Vector mul.
sadd.vv vv8, vv4, vv8  # Vector add.
addu t7, t4            # Scalar add.
    
```

Caltech CS184 Spring2005 -- DeHon

46

## T0 Execution Example



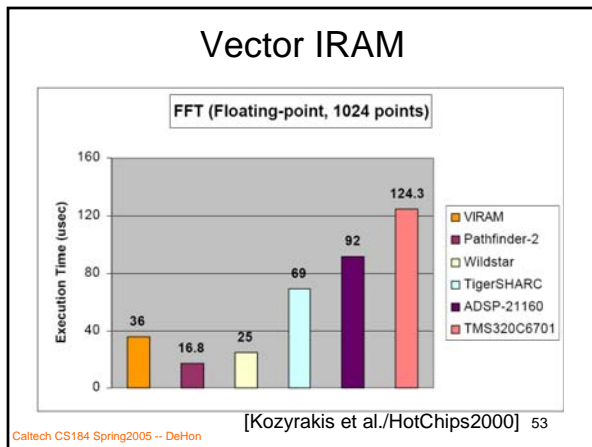
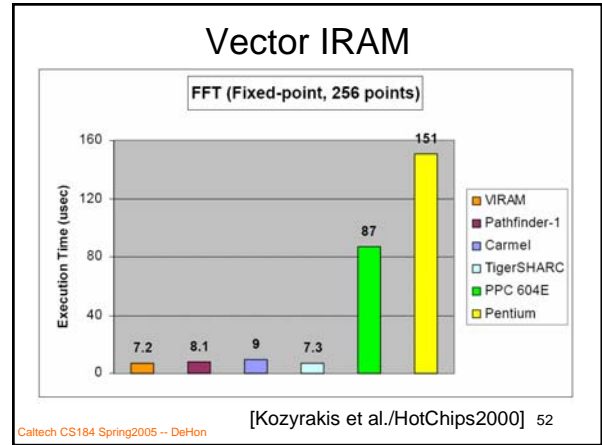
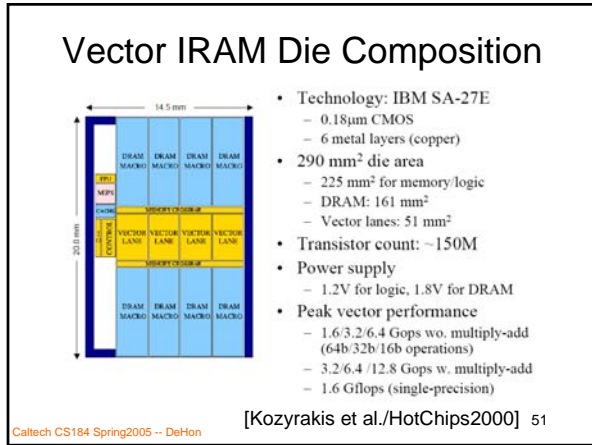
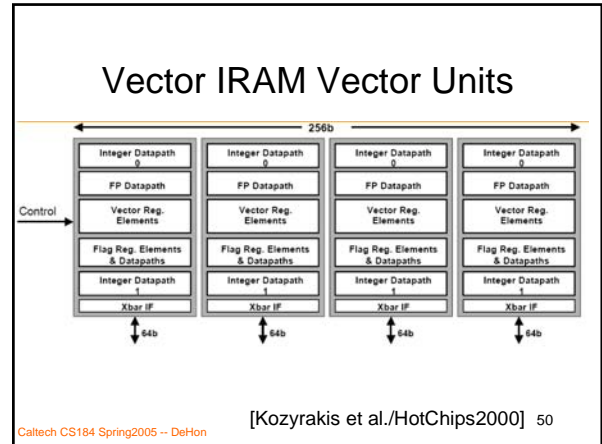
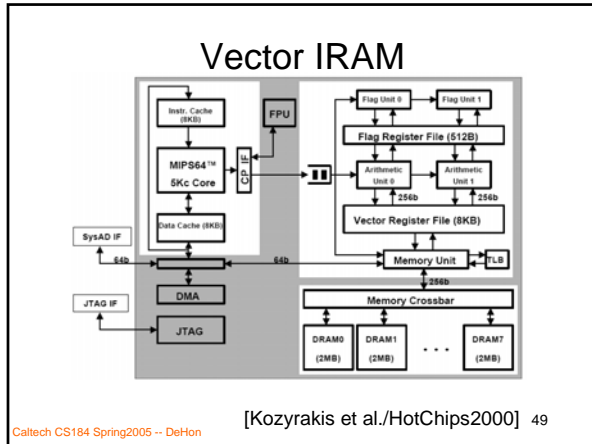
Calto

## T0: Vector Microprocessor

- Higher raw density than (super)scalar microprocessors
  - 22 ALU Bit Ops/ $\lambda^2$ s (vs. <10)
- Clean ISA, scaling
  - contrast VIS, MMX
- Easy integration with existing  $\mu$ P/tools
  - assembly library for vector/matrix ops
  - leverage work in vectorizing compilers

Caltech CS184 Spring2005 -- DeHon

48



### NEC EARTH Simulator

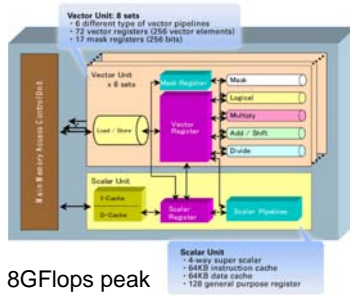
- Top Ranked Computer in the World
  - June 2002 → June 2004
- 640 vector processing chips

<http://www.es.jams-tec.go.jp/esc/eng/publications/images/ES.pdf>

[Kozyrakis et al./HotChips2000] 54

Caltech CS184 Spring2005 -- DeHon

## NEC Arithmetic Processor



- 500MHz clock, 8GFlops peak

<http://www.es.jamstec.go.jp/esc/eng/ES/hardware.html>

Caltech CS184 Spring2005 -- DeHon

55

## Predecessor

- Cray 1, 2, 3
- Held records for fastest supercomputer long before



Caltech CS184 Spring2005 -- DeHon

56

## More from Cray



Cray2



Cray XMP

Caltech CS184 Spring2005 -- DeHon

57

## Admin

Caltech CS184 Spring2005 -- DeHon

58

## Project

- P567
  - Is out today
  - Software Infrastructure session with Rafi in evening?
    - 5pm?

Caltech CS184 Spring2005 -- DeHon

59

## Big Ideas

- Model for computation
  - enables programmer think about machine capabilities a high level
  - abstract out implementation details
  - allow scaling/different implementations
- Exploit structure in computation
  - use to reduce hardware costs
- Vector/SIMD – simple model, admits dense implementations
  - How much fits into model?

Caltech CS184 Spring2005 -- DeHon

60