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

Architecture Instruction Taxonomy

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

How do an *if*?

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

Branchless Multiply Example

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

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

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

ABS with Mask

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

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

Synchronization

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

Communications

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

Communications

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

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

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
…not always that simple…

- Often, the trick to making vector models 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.

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.

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.

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 is good enough.
  - Detect when all processors done.

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.
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)

Contrast VLIW/SS

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

Resources
Peak Densities from Model

- Only 2 of 4 parameters
  - small slice of space
  - 100× density across

- Large difference in peak densities
  - large design space!

Calibrate Model

- FPGA model: \( w = 1, d = c = 1, k = 4 \)
  - \( 880K_{12} \)
- Xilinx 4K
  - \( 630K_{12} \)
- Altera 8K
  - \( 930K_{12} \)

- SIMD model: \( w = 1000, c = 0, d = 64, k = 3 \)
  - Abacus
    - \( 170K_{12} \)
    - \( 190K_{12} \)

- Processor model: \( w = 32, d = 32, c = 1024, k = 2 \)
  - MIPS-X
    - \( 2.6M_{12} \)
  - \( 2.1M_{32} \)

Examples

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]
Abacus: Addition

\[ A = \begin{bmatrix} 4 & 1 & 7 & 3 & 2 & 1 & 5 \end{bmatrix} \]
\[ +\text{-scan} = \begin{bmatrix} 4 & 5 & 12 & 20 & 23 & 25 & 26 & 31 \end{bmatrix} \]
\[ \text{max-scan} = \begin{bmatrix} 4 & 4 & 7 & 8 & 8 & 8 & 8 & 8 \end{bmatrix} \]
\[ +\text{-reduce} = 31 \]

Abacus: bit-wise SIMD

- High raw density:
  - 660 ALU Bit Ops/λ²s
  - Compare peak of 10 bops/λ²s for proc.
  - Compare ~100 bops/λ²s for FPGAs
- Do have to synthesize many things out of several operations
- Nearest neighbor communication only

Abacus: Cycles

<table>
<thead>
<tr>
<th>Operation</th>
<th>8-bit</th>
<th>16-bit</th>
<th>32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cycles</td>
<td>GOPS</td>
<td>Cycles</td>
</tr>
<tr>
<td>Add</td>
<td>4</td>
<td>4.8</td>
<td>4</td>
</tr>
<tr>
<td>Shift</td>
<td>2</td>
<td>8.0</td>
<td>2</td>
</tr>
<tr>
<td>Accumulate</td>
<td>3</td>
<td>5.2</td>
<td>3</td>
</tr>
<tr>
<td>Move</td>
<td>3</td>
<td>5.2</td>
<td>4</td>
</tr>
<tr>
<td>Compare</td>
<td>6</td>
<td>2.6</td>
<td>11</td>
</tr>
<tr>
<td>Multiply (16 x 16)</td>
<td>188</td>
<td>0.03</td>
<td></td>
</tr>
</tbody>
</table>

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]
Vector Scaling

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

Just dependence vector-op to vector-op

T0 Microarchitecture

T0 Pipeline

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.

T0 Execution Example

T0: Vector Microprocessor

• Higher raw density than (super)scalar microprocessors
  – 22 ALU Bit Ops/2s (vs. <10)
• Clean ISA, scaling
  – contrast VIS, MMX
• Easy integration with existing µP/tools
  – assembly library for vector/matrix ops
  – leverage work in vectorizing compilers
Vector IRAM

Vector IRAM Vector Units

Vector IRAM Die Composition

Vector IRAM

NEC EARTH Simulator

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

NEC Arithmetic Processor

- 500MHz clock, 8GFlops peak

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

Predecessor

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

More from Cray

Cray2

Cray XMP

Admin

Big Ideas

- Model for computation
  - enables programmer think about machine capabilities at 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?

Project

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