CS184b: Computer Architecture (Abstractions and Optimizations)

Day 6: April 18, 2003
Statically Compiled ILP
VLIW, EPIC

Today

• Trace Scheduling
• VLIW uArch
• Evidence for
• What it doesn’t address
• EPIC: next generation of VLIW?
Problem

- Parallelism in Basic Block is limited
  - (recall average branch freq. Every 7-8 instrs)

Solution: Trace Scheduling

- Schedule likely sequences of code through branches
  - instrument code
    - capture execution frequency / branch probabilities
  - pick most common path through code
  - schedule as if that happens
  - add “patchup” code to handle uncommon case where exit trace
  - repeat for next most common case until done
Solution Validity

• Recall from Fisher/Predict paper
  – 50-150 instructions/mispredicted branch
Trace Example

• Bulldog Fig 4.2

Bulldog: A Compiler for VLIW Architectures
MIT Press 1986
ACM Doctoral Dissertation Award 1985

Trace Join Example

Bulldog p61

the code generator produced the following schedule:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c:=a*b</td>
<td>i:=i+1</td>
</tr>
<tr>
<td>2</td>
<td>d:=c-2</td>
<td>j:=i+2</td>
</tr>
</tbody>
</table>
### Trace Join Example

![Diagram of Trace Join Example](image1.png)

The code generator produced the following sequence:

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c := a + b</td>
</tr>
<tr>
<td>2</td>
<td>i := i + 1</td>
</tr>
<tr>
<td>3</td>
<td>d := c - 2</td>
</tr>
<tr>
<td>4</td>
<td>j := j + 2</td>
</tr>
</tbody>
</table>

Bulldog p61-62

### Trace Multi-Branch Example

![Diagram of Trace Multi-Branch Example](image2.png)

Bulldog p69
Trace Multi-Branch Example

Bulldog p69-70

Trace Advantage

- Avoid fragmentation
  - can’t fill issue slots because broken by branches
- Expose more parallelism
  - concurrent run things on different sides of branches
  - allow more global code motion (across branches)
Loops

• **Problem:** loops introduce (conditional) branches
  – Breaks up code to schedule
  – Adds overhead for testing
  – Maybe limited parallelism in single loop body

Loop Unrolling

• **Solution:** unroll the loop
  – Create larger basic block and trace-schedule
    • More stuff to work with
  – Loop less frequently
    • Amortize out loop control overhead
  – Common case will be many iterations of loop
Example

• i:=1
• LOOP
  IF i>n THEN EXITLOOP
  A[i]:=b[i]+c[i]
i:=i+1

Example Cont.

• i:=1
• LOOP
  If i>n then EXITLOOP
  A[i]:=b[i]+c[i]
i:=i+1
  If i>n then EXITLOOP
  A[i]:=b[i]+c[i]
i:=i+1
  If i>n then EXITLOOP
  A[i]:=b[i]+c[i]
i:=i+1
Example Cont.

- \(i := 1\)
- LOOP
  - If \(i > n\) then EXITLOOP
  - \(A[i] := b[i] + c[i]\)
  - \(i := i + 1\)
  - If \(i > n\) then EXITLOOP
  - \(A[i] := b[i] + c[i]\)
  - \(i := i + 1\)
  - If \(i > n\) then EXITLOOP
  - \(A[i] := b[i] + c[i]\)
  - \(i := i + 1\)

- Trace Schedule, Rename …
- \(i := 1, j = 2, k = 3\)
- LOOP
  - If \(i + 3 > n\) then CLEANUP
  - \(A[i] := b[i] + c[i]\)
  - \(i := i + 3\)
  - \(j := j + 3\)
  - \(A[k] := b[k] + c[k]\)
  - \(k := k + 3\)

- CLEANUP
  - …

Machine

- Single PC/thread of control
- Wide instructions

- Branching
- Register File
- Memory Banking
Branching

• Allow multiple branches per “Instruction”
  – n-way branch
• N-tests + 1 fall-through
  – order in trace order
  – take first to succeed
• Encoding
  – single base address
  – branch to base+i
    • i is test which succeeded

Split Register File

• Each cluster has own RF
  – (register bank)
  – can have limited read/write bw
• Limited networking between clusters
  – explicit moves between clusters when results needed elsewhere
Memory Banks

• Separate Memory Banks
  – dispatch set of non-conflicting loads/stores, each to separate memory banks
  – trick is can compiler determine non-conflict
    • (do layout o avoid conflicts)
  – has to know won’t conflict (for VLIW timing)

Memory Banks

• Avoid single memory bottleneck
• Avoid having to build n-ported memory
• Can make likelihood of conflict small
• Costs for crossbar between memory and consumers
• Arbitration required if can’t statically schedule access pattern
• Hotspots/poor bank allocation can degrade performance
ELI “Realistic”

Bulldog Fig 8.1

Ellis Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Speed-up</th>
<th>Input size</th>
<th>Unrolling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ideal</td>
<td>Real.</td>
<td>Ideal</td>
</tr>
<tr>
<td>MATMUL</td>
<td>25.5</td>
<td>7.4</td>
<td>50²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>FFT</td>
<td>48.3</td>
<td>6.9</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>SOLVE</td>
<td>18.9</td>
<td>6.2</td>
<td>128²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>SVD</td>
<td>16.2</td>
<td>5.4</td>
<td>30²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16/2</td>
</tr>
<tr>
<td>TRID1</td>
<td>2.7</td>
<td>.9</td>
<td>4096</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>TRID2</td>
<td>3.8</td>
<td>1.2</td>
<td>4096</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>TRID4</td>
<td>33.3</td>
<td>7.0</td>
<td>4096</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16/4</td>
</tr>
<tr>
<td>EOS</td>
<td>8.3</td>
<td>2.3</td>
<td>64²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>NEWRZ</td>
<td>19.8</td>
<td>7.6</td>
<td>60²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>QK61</td>
<td>10.2</td>
<td>4.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>QUANC8</td>
<td>8.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZEROIN</td>
<td>3.5</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>
Two CMOS VLIWs

- LIFE [ISSCC90] \(23 \text{ ALU bops/\lambda}^2\text{s}\)
- VIPER [JSSC93] \(9.8\)

What can/can’t it do?

- Multiple Issue?
- Renaming?
- Branch prediction?
  - Static
  - dynamic
- Tolerate variable latency?
  - Memory
  - functional units
Scaling

• Issue
• Bypass
• Register File
• N-way branch
• Memory Banking
• RF-RF datapath

Scaling

• Linear Scaling
  – Issue
  – Bypass (only within cluster)
  – Register File (separate per cluster)

• Super linear
  – Memory Banking \[ (\text{clusters})^2 ? \]
  – RF-RF datapath ?
    • Unclear from small examples (and didn’t study)
Scaling: N-way branch?

• Probably want to scale up branching with clusters (VLIW length)
• Use parallel prefix computation
  – depth goes as log(N)
  – area can be linear

Scaling: Thoughts

• W/ on-chip memory
  – banks local to clusters (distributed memory)
  – can schedule operations on clusters close to memory?
  – Communicate data among clusters (like RF to RF transfers) if need non-local
  – How much interconnect needed?
    • What’s the locality of data communication?
    • Recall interconnect richness study from last term
“Weaknesses”

- Binary Compatibility
  – lack thereof
- No “Architecture”
- Exceptions

VLIW Roundup

- Exploit ILP
- w/out all the hardware complexity and cost
- Relegate even more interesting stuff to the compiler (REMISC?)

  …but no binary compatibility path
Admin

• Question:
  – Familiar with software pipelining?

• Monday:
  – IA-64 (concrete realization of EPIC)
  – Binary Translation
    • (shuffle up from Friday)

Big Ideas

• Get better packing/performance scheduling large blocks
• Common case
• Feedback
  – (future like past)
  – discover common case
• Binding Time hoisting
  – Don’t do at runtime what you can do at compile time
• Stable abstraction