Control Independence in Trace Processors

Eric Rotenberg
Jim Smith

North Carolina State University
Department of Electrical and Computer Engineering
ericro@ece.ncsu.edu
Introduction

• High instruction-level parallelism (ILP)
  - requires large instruction scheduling window
  - branches make this difficult

• Branches introduce control dependences
  - must execute branch to fetch next instructions
  - correct branch prediction/speculation: essentially eliminates control dependences
  - mispredictions are still a major bottleneck: complete squashing
Introduction

• Solutions to branch misprediction bottleneck
  1. better branch prediction
  2. forms of multi-path execution
  3. predication
  4. control independence
Control Independence

- Incorrect control dependent instructions
- Correct control dependent instructions
- False data dependences
- True data dependences
- Re-convergent point
- Control independent instructions
Basic Requirements

• Basic requirements for misprediction recovery
  - Detect the re-convergent point
  - Insert/remove instructions from middle of window
  - Form correct data dependences
  - Selectively reissue instructions
Complexity

• Major source of complexity: sophisticated window management
  - Arbitrary insertion/removal of instructions from middle of window
  - Conventional instruction window (“reorder buffer”) is managed as a fifo
  - Arbitrary shifting, or linked-list implementation, difficult

• Conveying control dependence information
  - maybe don’t want software to convey and expose re-convergent points

• Repairing data dependences
Trace Processor

Processing Element 0

Global Registers

Local Registers

Func Units

Issue Buffers

Processing Element 1

Processing Element 2

Processing Element 3

Data Cache

Speculative State

Next Trace Predict

Trace Cache

Global Rename Maps

Live-in Value Predict

Eric Rotenberg
Micro-32

Control Independence in Trace Processors

Slide 7
• Three key ideas
  - Treating traces as fundamental unit of control flow enables sophisticated window management (hierarchy)
  - Trace selection ensures and identifies trace-level re-convergence
  - Existing support for data speculation is easily leveraged for selectively re-issuing incorrect-data dependent instr.
1. Hierarchical management of control flow
   - isolate intra-trace flow from inter-trace flow
   - branch and control independent point in same trace
   => fine-grain control independence (FGCI)
2. Hierarchical management of resources

- large unit of allocation (PE/trace): easy to manage PE allocation/de-allocation flexibly
- branch and control independent point in different traces

=> coarse-grain control independence (CGCI)
Trace-level Re-convergence

(a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

(b) T1
   1 2 3 4 5 6 7 8
   T1'
   1 2 3 4 5 6 13 14

   T2
   9 10 11 12 15 16 17 18
   T2'
   15 16 17 18 19 20 21 22

   T3
   19 20 21 22 23 24 25 26
   T3'
   23 24 25 26 27 28 29 30

   T4
   27 28 29 30 31 32 33 34

(c) T1
   6 7 8 9 10 11 12 15
   T2
   6 7 8 9 10 11 12 15
   T2'
   15 16 17 18 19 20 21 22

   T3
   19 20 21 22 23 24 25 26
   T4
   24 25 26 27 28 29 30 31

(d) T1
   1 2 3 4 5 6 7 8
   T1'
   1 2 3 4 5 6 13 14

   T2
   9 10 11 12 15 16 17 18
   T2'
   15 16 17 18 19 20 21 22

   T3
   19 20 21 22 23 24 25 26
   T4
   27 28 29 30 31 32 33 34

   global re-convergent point
FGCI Trace Selection

• FGCI-algorithm analyzes arbitrarily complex forward-branching regions
  - locates re-convergent point
  - computes longest control dependent path

• FGCI trace selection uses info to “pad” shorter paths

<table>
<thead>
<tr>
<th>trace</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A,B,C,F,H}</td>
<td>16</td>
</tr>
<tr>
<td>{A,B,D,F,H}</td>
<td>15</td>
</tr>
<tr>
<td>{A,E,F,H}</td>
<td>11</td>
</tr>
<tr>
<td>{A,E,G,H}</td>
<td>15</td>
</tr>
</tbody>
</table>

embeddable region

guaranteed stop point
Control Independence in Trace Processors

• Identify “global” control independent points
  - Imprecise: points don’t correspond directly with any particular branch
  - But, broad coverage of branch mispredictions

• CGCI trace selection terminates traces at return-points and not-taken backward branches (black dots below)
 Repairing Data Dependences

• Register dependences
  - Re-dispatch control independent traces

• Memory dependences
  Missspeculated loads due to control independence

  ==

  Missspeculated loads due to speculative disambiguation

• Selective re-issuing
  - Already supported for data speculation
Graph Labels

- **RET**
  - CGCI only
  - Use the return-point heuristic for all mispredictions

- **MLB-RET**
  - CGCI only
  - Try MLB first for mispredicted loop branches
  - RET covers all mispredicted branches

- **FG**
  - FGCI only

- **FG + MLB-RET**
Performance

% IPC improvement over base

-5% 0% 5% 10% 15% 20% 25%

gcc go comp jpeg xlisp

RET MLB-RET FG FG + MLB-RET
Conclusion

• Trace processor is a good microarchitecture for exploiting control independence.

• Concepts
  1. Hierarchy is a key enabler for exploiting control independence (sophisticated window management)
  2. FGCI and CGCI trace selection ensure and identify trace-level re-convergence
  3. Existing support for data speculation is easily leveraged

• Performance gains of up to 25%