Trace Based Switching For A Tightly Coupled Heterogeneous Core

Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, Scott Mahlke

Micro-46
December 2013
Outline

• Fine-grained heterogeneity

• Don’t React – Predict!

• Trace-Based Switching Controller

• Results

• Conclusion
Single-ISA Heterogeneity

Out-of-order
✓ Fast
  ✓ Good for high performance phases
✗ Power hungry structures
  ✗ Wastes energy on low performance phases

Energy savings with minimal performance loss

In-order
✗ Smaller/Slower
  ✗ 1/2 the performance of big
✓ Simpler, low power structures
  ✓ 3X more energy efficient
Traditional Heterogeneous Architectures

Transfer overhead = ~20K Cycles
Minimum switching interval = ~1M instructions

What about low performance phases at finer-granularity?
Performance Change In GCC

Huge performance changes within a coarse quantum!

Average IPC over 1M Instructions

- Average IPC over 1M Instructions
  - Over a 1M instruction window (Quantum)
  - Average IPC over 2K Quanta
  - Huge performance changes within a coarse quantum!
Fine-grain Heterogeneity Has Potential

Oracle – theoretical maximum per quantum size

(Subject to a maximum 5% performance loss target)

Finer is better!
Composite Cores

Transfer overhead = ~35 Cycles
Minimum switching interval = ~1000 instructions

*Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012
Outline

• Fine-grained heterogeneity
  • Don’t React – Predict!

• Trace-Based Switching Controller

• Results

• Conclusion
Traditional Switching Controller

Assumption:
A quantum’s behavior is indicated by the recent past
Reactive Follies at Fine-Granularity

Reactive Controller at 300 Instruction Quanta, gcc

Incorrectly picked little $\rightarrow$ lost performance, on 29% of quanta

Incorrectly picked big $\rightarrow$ lost opportunities for energy savings on 26% of quanta
Fine-granularity “Reacts” Badly

Don’t React – Predict!

% Time Spent On Little

# Instructions per Quantum
Outline

• Fine-grained heterogeneity

• Don’t React – Predict!

• Trace-Based Switching Controller

• Results

• Conclusion
History Based Prediction

**Observation**
Super-Trace
Code repeats (loops, functions)
Exploit this inherent repeatable nature of code

**Objective**
Behavior placed in program context
1. Predict oncoming Super-traces
2. Predict its backend preference

**Our Assumption**
A Super-Trace’s behavior history follows a pattern
Super-trace Construction

- **Backedge 0**
- **Backedge 1**
- **Backedge 2**
- **Backedge 3**

BB7

Function call

BB1

BB2

BB3

BB4

BB5

BB6

BB7

BB1

BB2

BB3

BB4

BB5

BB7

Function call

Observe dynamic execution after a function call.
Super-trace Construction

Trace: Block of code defined at backedge boundaries

Performance highly dependent on context

Combine traces to meet minimum length requirement (300 instructions)
Super-trace Construction

Count instructions between backedges

Backedge0

Backedge1

Backedge2

Backedge3

Super-Trace!

ID = BE1 + BE2 + BE3
Trace-Based Controller Overview

Two-level predictor tables

1. Super-Trace Constructor

2. Next Super-Trace Predictor

3. Backend Predictor

4. Feedback to Predictors*

Super-Trace ID

Next Super-Trace ID

Update

Run on Big/Little?

Big Backend

Little Backend

Committed Instructions & Backedge PCs

*Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012
1. **Super-Trace Constructor**

1. Check for minimum instruction length constraint

2. Hash backedge PCs to provide a super-trace ID
1 Super-Trace Constructor

Hash backedge PCs to provide a super-trace ID

<table>
<thead>
<tr>
<th>BE12</th>
<th>BE11</th>
<th>BE10</th>
<th>BE9</th>
<th>BE8</th>
<th>BE7</th>
<th>BE6</th>
<th>BE5</th>
<th>BE4</th>
<th>BE3</th>
<th>BE2</th>
<th>BE1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

More Recently seen

9 (To 2)

9 bit STrace ID
## Next Super-Trace Predictor

<table>
<thead>
<tr>
<th>1. Super-Trace ID</th>
<th>Head</th>
<th>Replaceability</th>
<th>2. Super-Trace ID</th>
<th>Head</th>
<th>Replaceability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9 bits)</td>
<td>(3 bits)</td>
<td>(2 bits)</td>
<td>(9 bits)</td>
<td>(3 bits)</td>
<td>(2 bits)</td>
</tr>
<tr>
<td><strong>S-Trace Id\textsubscript{i+1}</strong></td>
<td><strong>Head\textsubscript{i+1}</strong></td>
<td>10</td>
<td><strong>S-Trace Id\textsubscript{m+1}</strong></td>
<td><strong>Head\textsubscript{m+1}</strong></td>
<td>11</td>
</tr>
</tbody>
</table>

(From 1)  

(To 3)
Next Backend Predictor

(From 2 )

Strace ID\textsubscript{i+1}

Big/ Little Confidence

(2 bits)

10

Run on Big!

Big Backend

Little Backend
# Evaluation

<table>
<thead>
<tr>
<th>Architectural Feature</th>
<th>Parameters</th>
</tr>
</thead>
</table>
| Big Core              | 3 wide O3 @ 1.2GHz  
12 stage pipeline  
128 ROB Entries  
128 entry register file |
| Little Core           | 2 wide InOrder @ 1.2GHz  
8 stage pipeline  
32 entry register file |
| Memory System         | 32 KB L1 i/d cache, 2 cycle access  
1MB L2 cache, 15 cycle access  
1GB Main Mem, 80 cycle access |
| Simulator             | Gem5, Full system |
| Energy Model          | McPAT |
| Benchmarks            | SPEC 2k6, compiled for Alpha ISA  
Fast Forward 2 billion instructions,  
simulate 100 million instructions |
Outline

• Fine-grained heterogeneity
• Don’t React – Predict!
• Trace-Based Switching Controller
• Results
• Conclusion
Oracle

SuperTrace-Based Controller

Big Quantums = Red, Little Quantums = Blue

Easy to predict:
Distinct and repeatable phase behavior

Relative Performance of Big vs Little

Super-traces in Increasing Time Order

h264ref

0  50000  100000  150000  200000  250000  300000
Oracle

Reactive Controller

SuperTrace-Based Controller

Hard to predict:
Highly variable data & control flow

Big Quantsums = Red,
Little Quantsums = Blue

Relative Performance of Big vs Little

Super-traces in Increasing Time Order

gcc
Time spent on the Little backend increases by 46% on average.
Performance Compared To Baseline OoO

The controllers all honor the 95% performance target
On average, we conserve 43% more energy than a reactive controller.
Conclusion

• Don’t React – Predict!
  – Utilize Little 46% more than current work

• 15% energy savings over the baseline (43% more than existing work)
  – small hardware overheads (1.9kB)
Trace Based Switching For A Tightly Coupled Heterogeneous Core

Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, Scott Mahlke

shrupad@umich.edu

Thank you!

Micro-45
December 2013
Backup
Case for Heterogeneous Cores

- High energy usage
- Yields high performance

High performance cores waste energy on low performance phases

- Large structures (ROB, Rename Table, LSQ)
- Higher issue width

Run low performance phases on slower, but energy-efficient cores

General purpose application executing on a high performance core
Core Energy Comparison

Out-of-Order
- Instruction Fetch: 22%
- Branch Target Buffer: 5%
- Register Alias Table: 6%
- Reservation Stations: 8%
- Reorder Buffer: 11%
- Integer Execution Unit: 15%
- Data Cache Unit: 11%
- Floating Point Execution Unit: 8%
- Memory Order Buffer: 6%
- Global Clock: 8%

In-Order
- Fetch: 37%
- Decode: 18%
- Issue: 14%
- Execute: 9%
- Memory: 15%
- Writeback: 7%

Do we always need the extra hardware?

Brooks, ISCA'00
Dally, IEEE Computer'08
Reactive Online Controller

User-Selected Performance

Threshold Controller

Switching Controller

Little Model

Big Model

Little uEngine

Big uEngine

\[ S^* \sum CPI_{\text{Big}} \]

\[ \Delta CPI_{\text{Threshold}} \leq CPI_{\text{big}} \]

\[ \Delta CPI_{\text{Target}} \]

\[ CPI_{\text{error}} \]

\[ CPI_{\text{Actual}} \]

\[ \sum CPI_{\text{Observed}} \]

\[ CPI_{\text{little}} \leq CPI_{\text{big}} \]
Accuracy

![Accuracy Graph]

- Reactive
- PerfectNextTraceKnowledge
- SuperTrace
- BackedgeFrequency

% Quantaums Accurately Mapped vs Quantum Size In Instructions
Sensitivity to Other Schemes

![Bar chart showing sensitivity to other schemes for different benchmarks. The x-axis represents different benchmarks, while the y-axis represents the percentage of quanta accurately mapped. The chart compares our scheme to 2LevelAdaptive-Local and 2LevelAdaptive-Global schemes.](image-url)
Composite Cores – Controller Overview

Performance Monitor (Feedback Controller)

Core Performance Models (RegressionModel)

Backend selector

Performance Loss < Threshold

Yes

No

Little backend

Big backend

Estimated target performance

Estimated big & little performance difference

Threshold

Observed performance metrics

Little backend

Big backend

Threshold

Yes

No
Next Super-Trace Predictor

<table>
<thead>
<tr>
<th>1. S-Trace ID</th>
<th>Head</th>
<th>Confidence</th>
<th>2. S-Trace ID</th>
<th>Head</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9 bits)</td>
<td>(3 bits)</td>
<td>(2 bits)</td>
<td>(9 bits)</td>
<td>(3 bits)</td>
<td>(2 bits)</td>
</tr>
</tbody>
</table>

S-Trace Id_{i+1} Head_{i+1} 11 S-Trace Id_{k+1} Head_{k+1} 00

Update – Incorrect Decision

(Strace Idj) (Headj+1) (Strace Idj+1) (Strace Idi+1)

(From AcYve Backend) (From Active Backend)
Huge performance changes within a coarse quantum!
Fine-grain Performance Phases Exist

**Fine grain offers more opportunity to save energy by exploiting:**

- Dependent compute
- Dependent load misses
- Branch mispredicts

*Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012*
## Next Super-Trace Predictor

### Table

<table>
<thead>
<tr>
<th>1. Super-Trace ID</th>
<th>Head</th>
<th>Replacability</th>
<th>2. Super-Trace ID</th>
<th>Head</th>
<th>Replacability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9 bits)</td>
<td>(2 bits)</td>
<td>(2 bits)</td>
<td>(9 bits)</td>
<td>(3 bits)</td>
<td>(2 bits)</td>
</tr>
<tr>
<td>STrace Id_{i+1}</td>
<td>Head_{i+1}</td>
<td>10</td>
<td>S-Trace Id_{m+1}</td>
<td>Head_{m+1}</td>
<td>11</td>
</tr>
</tbody>
</table>

### Diagram

- **Update:** Correct Decision
- **Update:**
- **(From 1) (To 3)**

(From Active Backend)
Next Backend Predictor

(From \( i+1 \))

- Strace ID\(_{i+1}\)
- Observed Strace ID\(_{i+1}\)
- Big Backend
- Little Backend
- Big/Little (2 bits)
- 10
- Update: Correct Decision
- Performance Counters
- Observed Performance Metrics
- Feedback Generator
Fine-grained Heterogeneous Architectures

(B) H3 *

*Rationale for a 3D Heterogeneous Multi-core Processor, Rotenberg, ICCD 2013