Energy-Efficient Speculative Threads:
Dynamic Thread Allocation in Same-ISA Heterogeneous Multicore System

Yangchun Luo*, Venkatesan Packirisamy†,
Wei-Chung Hsu‡, and Antonia Zhai*

*University of Minnesota – Twin Cities
† NVIDIA Corporation
‡ National Chiao Tung University, Taiwan
Background

Traditional Program

Multicore Processor

Multicore requires thread-level parallelism
Speculative Parallelism

Sequential

Store *p
Load *q

Traditional Parallelization

Store *p
Load *q

p != q ??

Thread-Level Speculation (TLS)

p != q

Store 88
Load 20
Parallel execution

p == q

Store 88
Load 88
Speculation Failure

More potential parallelism
Speculation vs. Energy Efficiency

**Successful Speculation**
- $p \neq q$
- Improve performance
- Reduce leakage duration

**Failed Speculation**
- $p = q$
- Waste dynamic power
- More leaking component

Can we exploit performance without compromising energy efficiency?
Impact from Underlying Hardware

TLS Energy Efficiency vs. Hardware Configuration
[Packirisamy ICCD’08]

SMT Architecture

CMP Architecture

Overall higher efficiency
Better in some cases
Optimization Opportunities

Resource contention
- Failed threads competes
  - Use CMP

Low instr. level parallelism
  - Use simpler cores
  - TLS exploits both ILP & TLP

Unique cache patterns of TLS
- Multiple cache activated
  - Use smaller caches
Our Proposal

Program Execution

Underlying Hardware

On-chip Heterogeneity

Dynamic Resource Allocation
Same-ISA Heterogeneity

(1) Multi-Threading Execution Mode

(2) Core Computing Power (issue width/SMT support)

(3) L1 Cache Size (set/associativity)

What components to integrate?
Design Space Exploration

An Unbounded Heterogeneous Multicore

Core Parameters | 2-issue | 4-issue
--- | --- | ---
Fetch/Issue/Commit Width | 4/2/2 | 6/4/4
ROB/LSQ Entries | 64/32 | 128/64
Integer/FP Units | 2/3 | 4/4

No Power-On and Off Overheads
No Cache Warm-up Cost
Component Usage

Sequential Segments

- Others
- SEQ_4iss_64K4w
- SEQ_4iss_16K4w
- SEQ_2iss_16K4w

Parallel Segments

- CMP
- SMT
- 2-iss
- 4-iss
- 6-iss
- 16K
- 32K
- 64K

Always favor 4-way set associative cache
Proposed Integration

<table>
<thead>
<tr>
<th>CMP</th>
<th>SMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-iss</td>
<td>4-iss</td>
</tr>
<tr>
<td>16K</td>
<td>32K</td>
</tr>
</tbody>
</table>

4-way set associativity

4-issue SMT

2-issue non-SMT

64K 4-way L1

Resizable by Sets

How much improvement?

University of Minnesota
Improvement Estimation

No overheads & Oracle thread allocation

Baseline Sequential Program

SMT

Total Improvement Upperbound: 29%
Improvement Estimation
No overheads & Oracle thread allocation

Proposal: 29% Improvement over SMT
Unbounded: 33% Improvement over SMT

Our proposal captures most of the benefit!
Overhead Sources

1. Startup Overhead

Startup

Powered-off

Static power consumed

2. Cache Reconfiguration Overhead

Bigger size

Smaller size

Dirty lines

Write back to L2

Smaller size

Mapping changed

Content discarded

Bigger size

cold
Overhead Impacts

Oracle thread allocation

Heterogeneous vs. Homogeneous

Heterogeneous vs. Homogeneous

Overheads

-80%

+29%

University of Minnesota
Overhead Mitigation

**Benchmark Statistics**
- Average < 300 instr./thread
- Overall coverage ≈ 75%

**Throttling Mechanisms**
- Reduce reconfiguration frequency
  - Only when duration > overhead
- Delay device powering-off

**Fine Granularity**

![Graph showing benchmark statistics and throttling mechanisms](graph.png)
Determining Resource Configuration

The Difficulty: Intertwined Factors

Our Solution: **Divide & Conquer**

- Multithread Type: SMT vs. CMP
- Core Issue Width: 2-issue vs. 4-issue
- L1 Cache Size: 64KB vs. 16KB

**Runtime Monitoring**
- Hardware Performance Counters
- Sampling Run: 4-issue SMT with 64K L1
Decision Makings

Hardware Performance Monitor

Cycles non-speculative thread stall due to resource contention

IPC

Reused cache blocks

Normalized ED2P

Oracle Allocator
Realistic Allocator

Right Decisions!
Comparisons

Sequential (SEQ)

Unbounded

SMT

Sequential Program
Norm. Baseline

Heterogeneous
Heterogeneous vs. Homogeneous

- Heterogeneous
- Homogeneous

Normalized ED2P

- SMT
- Heterogeneous

4% higher perf.
6% less energy

ED²P Improvement wrt. SMT

Heterogeneity is beneficial
Execution Mode Breakdown

Thread Migration

- 4 issue
- 2 issue
- 4 issue

Cache Resizing

- 16K
- 64K
- 16K

Coverage

- Dynamic allocation is essential

Start-up

- SMT_4iss_16K
- SMT_4iss_64K
- CMP_2iss_16K
- CMP_4iss_16K
- SEQ_2iss_16K
- SEQ_2iss_64K
- SEQ_4iss_16K
- SEQ_4iss_64K
Comparisons

Sequential (SEQ)

SMT

Sequential Program
Norm. Baseline

Unbounded

Heterogeneous
Heterogeneous vs. Sequential

ED²P Compared to SEQ

Baseline

Baseline

-54%

38% higher perf.
7% more energy

Improve Performance Efficiently
| Related work | | | | |
|---|---|---|---|
| TLS Energy Efficiency and Hardware Configuration | CMP and SMT favored differently  
→ Heterogeneous integration | | |
| [Packirisamy, Luo, Zhai et al ICCD’08] | | | |
| Energy-Efficient TLS on a CMP | Ours: matching threads with configuration  
Theirs: can complement our system | | |
| [Renau et al ICS’05] | | | |
| Same-ISA Heterogeneous Multicore | Ours is different:  
▪ Speculative threads  
▪ Fine granularity and overhead mitigation | | |
| [Kumar et al Micro’03]  
[Kumar et al ISCA’04] | | | |
| Dynamic Perf. Tuning for TLS | Integrate to extract efficient threads | | |
| [Luo, Packirisamy, Zhai et al ISCA’09] | | | |
Conclusion

Evaluation Summary:

- **44%** better than uniprocessor
- **13%** better than homogeneous