LeadOut: Composing Low-Overhead Frequency-Enhancing Techniques for Single-Thread Performance in Configurable Multicores

Brian Greskamp, Ulya Karpuzcu, Josep Torrellas

http://iacoma.cs.uiuc.edu/
Per-Thread Performance Trend

Top SPECint92 Score

Year

Historical

Ulya Karpuzcu
Per-Thread Performance Trend

Top SPECint92 Score vs. Year

Historical

Contemporary

Ulya Karpuzcu
Near-Future CMP Environment
Near-Future CMP Environment

Intel Larrabee Performance [SIGGRAPH 2008]

- Production Fluid
- Production Cloth
- Marching Cubes
- Sports Video Analysis
- Text Indexing
- Foreground Estimation
- Human Body Tracking
- Portfolio Management
- 3D-FFT
- BLAS3
Near-Future CMP Environment

- Throughput applications demand more cores

**Intel Larrabee Performance [SIGGRAPH 2008]**

```
Parallel Speedup
```

```
Larrabee Units (1GHz Cores)
```

- Production Fluid
- Production Cloth
- Marching Cubes
- Sports Video Analysis
- Text Indexing
- Foreground Estimation
- Human Body Tracking
- Portfolio Management
- 3D-FFT
- BLAS3
Near-Future CMP Environment

- Throughput applications demand more cores
- Sequential applications need fast cores

Intel Larrabee Performance [SIGGRAPH 2008]

[Graph showing speedup vs. Larrabee units]
Near-Future CMP Environment

- Throughput applications demand **more** cores
- Sequential applications need **fast** cores
- Amdahl’s Law: Most applications need some fast cores
Near-Future CMP Environment

• Throughput applications demand more cores

• Sequential applications need fast cores

• Amdahl’s Law: Most applications need some fast cores

How to get faster cores without compromising core count?
CMP Design Space
CMP Design Space

- No compromise in core count
CMP Design Space

• No compromise in core count
  ✓ Throughput performance not affected
CMP Design Space

- No compromise in core count
  ✓ Throughput performance not affected
- Per-thread acceleration is configurable at runtime
CMP Design Space

• No compromise in core count
  ✓ Throughput performance not affected
• Per-thread acceleration is configurable at runtime
  ✓ Pay for per-thread performance on demand
CMP Design Space

• No compromise in core count
  ✓ Throughput performance not affected
• Per-thread acceleration is configurable at runtime
  ✓ Pay for per-thread performance on demand
• Minimal core design effort
CMP Design Space
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
</tr>
<tr>
<td>Core Fusion</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓ -</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
### CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓ -</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>V/f boosting</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes
- ✓: Yes
- ✗: No
- ✗: Don't care
- -: Inapplicable
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>V/f boosting</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Legend: ✓ = Yes, ✗ = No, - = Partial*
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>×</td>
<td>✓−</td>
<td>×</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>V/f boosting</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
### CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>V/f boosting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
# CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous CMPs</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Core Fusion</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Thread Level Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>V/f boosting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## CMP Design Space

<table>
<thead>
<tr>
<th>Approach</th>
<th>No Compromise in # Cores</th>
<th>Speed-up on Demand</th>
<th>Minimal Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut-down idle cores to run performance critical threads at higher than nominal frequencies</td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
</tr>
<tr>
<td>Configurable Timing Speculation</td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
</tr>
<tr>
<td>V/f boosting</td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
<td><img src="https://i.imgur.com/Ok.png" alt="Checkmark" /></td>
</tr>
</tbody>
</table>
Contribution: LeadOut
Contribution: LeadOut

In a large multicore with varying number of busy cores...
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
- Each alone is unable to bring the multicore all the way up to its P/T envelope
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
  - Each alone is unable to bring the multicore all the way up to its P/T envelope
  - Some of the available P/T headroom remains untapped
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
  - Each alone is unable to bring the multicore all the way up to its P/T envelope
  - Some of the available P/T headroom remains untapped
- TS and V/f boosting are complementary
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
  - Each alone is unable to bring the multicore all the way up to its P/T envelope
  - Some of the available P/T headroom remains untapped
- TS and V/f boosting are complementary
  - Bounded by different constraints
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
  - Each alone is unable to bring the multicore all the way up to its P/T envelope
  - Some of the available P/T headroom remains untapped
- TS and V/f boosting are complementary
  - Bounded by different constraints
  - Can be synergistically combined
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
  - Each alone is unable to bring the multicore all the way up to its P/T envelope
  - Some of the available P/T headroom remains untapped
- TS and V/f boosting are complementary
  - Bounded by different constraints
  - Can be synergistically combined
  - Speed-ups for single thread performance multiply
Contribution: LeadOut

In a large multicore with varying number of busy cores...

- Individual application of TS or V/f Boosting is suboptimal
- Each alone is unable to bring the multicore all the way up to its P/T envelope
- Some of the available P/T headroom remains untapped
- TS and V/f boosting are complementary
- Bounded by different constraints
- Can be synergistically combined
- Speed-ups for single thread performance multiply

Goal: Apply the two techniques until the cores reach their maximum allowed power or temperature
Timing Speculation (TS)
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$. 
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$.
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$

- Increase $f$ at constant $V$
Timing Speculation (TS)

Boost core frequency \( f \) beyond nominal limits at **constant** supply voltage \( V \)

- Increase \( f \) at constant \( V \) → some paths **overshoot**
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$

- Increase $f$ at constant $V \rightarrow$ some paths **overshoot**
- Support for error detection and correction
Timing Speculation (TS)
Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$

- Increase $f$ at constant $V$ $\rightarrow$ some paths **overshoot**
- Support for error detection and correction
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$

Assuming as much $V/f$ headroom as possible, $P_E$ limits performance gain

- Increase $f$ at constant $V \rightarrow$ some paths overshoot
- Support for error detection and correction
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$. 
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at **constant** supply voltage $V$

- Cores can be homogeneous
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$

- Cores can be homogeneous
  - ✓ No compromise in core count
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$

- Cores can be homogeneous
  - ✓ No compromise in core count
  - ✓ Simple core design
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$

- Cores can be homogeneous
  - ✓ No compromise in core count
  - ✓ Simple core design

- Frequency-boosting is configurable at runtime
Timing Speculation (TS)

Boost core frequency $f$ beyond nominal limits at constant supply voltage $V$

- Cores can be homogeneous
  - ✓ No compromise in core count
  - ✓ Simple core design
- Frequency-boosting is configurable at runtime
  - ✓ Pay for per-thread performance on demand
Example Architecture [PACT07]: Paceline

Many-Core CMP
Example Architecture [PACT07]: **Paceline**

OS sees: One core

**Leader**

**Checker**
Example Architecture [PACT07]: Paceline

OS sees: One core

Critical Thread

Leader

Checker

Ulya Karpuzcu
Example Architecture [PACT07]: **Paceline**

OS sees: One core

**Leader**

**Critical Thread**

**Checker**
Example Architecture [PACT07]: Paceline

Leader

Checker
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Hints

Rated Clock

Checker
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker

Hints

Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker

Hints

Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker

Hints
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

= ?

= ?

Rated Clock

Checker

Hints
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Checker

Rated Clock

Hints

Example Architecture [PACT07]: Paceline
Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker

= ?

Hints

Example Architecture [PACT07]: Paceline

Speculative Clock

Leader

Rated Clock

Checker

= ?

Hints
Example Architecture [PACT07]: Paceline

Rated Clock

OS sees: Two cores

Thread 0

= ?

Hints

Thread 1

Rated Clock
Voltage-Frequency Boosting
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V$.
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

$P_E$ vs. $Freq$
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

Assuming as much $V/f$ headroom as possible, $V$ limits performance gain
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

- Cores can be homogeneous
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

- Cores can be homogeneous
  - ✓ No compromise in core count
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

- Cores can be homogeneous
  - No compromise in core count
  - Simple core design
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V \rightarrow$ No timing errors ($P_E = 0$)

- Cores can be homogeneous
  - ✓ No compromise in core count
  - ✓ Simple core design
- V/f-boosting is configurable at runtime
Voltage-Frequency Boosting

Boost core frequency $f$ beyond nominal limits by increasing $V$ $\rightarrow$ No timing errors ($P_E = 0$)

- Cores can be homogeneous
  - ✓ No compromise in core count
  - ✓ Simple core design
- V/f-boosting is configurable at runtime
  - ✓ Pay for per-thread performance on demand
Voltage-Frequency Boosting
Voltage-Frequency Boosting

Normalized Frequency

Normalized Power

- Dynamic Power
- Static Power

Normalized Frequency

1.00 1.05 1.10 1.15 1.20 1.25

0.0 0.5 1.0 1.5 2.0 2.5
Voltage-Frequency Boosting

Boosting V above nominal gives additional f increase at large power cost
Voltage-Frequency Boosting

Boosting V above nominal gives additional f increase at large power cost.

... tolerable if the chip is not fully loaded [Intel Turbo Boost]
Example: Intel Turbo Boost

Intel Turbo Boost [Rajesh Kumar and Pat Gelsinger, Fall 2008 IDF]
Example: Intel Turbo Boost

Intel Turbo Boost [Rajesh Kumar and Pat Gelsinger, Fall 2008 IDF]
Example: Intel Turbo Boost

Intel Turbo Boost [Rajesh Kumar and Pat Gelsinger, Fall 2008 IDF]
Example: Intel Turbo Boost

Lightly Threaded Workload

All active cores change Vdd/f at the same time
Managing Thermal Imbalance
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
- Configurable TS such as Paceline
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
- Configurable TS such as Paceline
  - Periodically swap leader and checker
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
- Configurable TS such as Paceline
  - Periodically swap leader and checker
- V/f Boosting
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
- Configurable TS such as Paceline
  - Periodically swap leader and checker
- V/f Boosting
  - In general, expected to reach $P/T$ limits at lower $f$
Managing Thermal Imbalance

Induced by boosting core frequency $f$ beyond nominal...

- Avoiding formation of hot spots by activity migration
  - Spread heat by transporting computation to a different location on die
- Configurable TS such as Paceline
  - Periodically swap leader and checker
- $V/f$ Boosting
  - In general, expected to reach $P/T$ limits at lower $f$
  - Move critical thread among two available cores
Composing the Techniques
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

• Suppose as much P/T headroom as possible
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

- Suppose as much P/T headroom as possible
- Paceline limited by $P_{E_{\text{MAX}}}$
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

• Suppose as much P/T headroom as possible

• Paceline limited by $P_{E_{MAX}}$
  
  • Performance drops due frequency of recovery past $P_{E_{MAX}}$
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

- Suppose as much P/T headroom as possible
- Paceline limited by $P_{E_{\text{MAX}}}$
  - Performance drops due frequency of recovery past $P_{E_{\text{MAX}}}$
- V/f Boosting limited by $V_{\text{MAX}}$
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

• Suppose as much P/T headroom as possible

• Paceline limited by $P_{E_{MAX}}$
  • Performance drops due frequency of recovery past $P_{E_{MAX}}$

• V/f Boosting limited by $V_{MAX}$
  • Devices become unreliable past $V_{MAX}$
Composing the Techniques

The techniques are complementary since each is limited by a different constraint...

- Suppose as much P/T headroom as possible

The techniques are orthogonal

- Performance drops due frequency of recovery past \( P_{E_{\text{MAX}}} \)
- V/f Boosting limited by \( V_{\text{MAX}} \)
- Devices become unreliable past \( V_{\text{MAX}} \)
Composing the Techniques
Composing the Techniques

Want to exploit all available P, T headroom
## Composing the Techniques

<table>
<thead>
<tr>
<th>Multicore Loading Condition</th>
<th>Bounding Constraints</th>
<th>Gain from combining?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBo(+Mig)</td>
<td>Paceline</td>
</tr>
<tr>
<td>T/P</td>
<td>V</td>
<td>T/P</td>
</tr>
<tr>
<td>Very High</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High to Moderate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Low</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Want to exploit all available P, T headroom
## Composing the Techniques

<table>
<thead>
<tr>
<th>Multicore Loading Condition</th>
<th>Bounding Constraints</th>
<th>Gain from combining?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBo(+Mig)</td>
<td>Paceline</td>
</tr>
<tr>
<td>T/P</td>
<td>V</td>
<td>T/P</td>
</tr>
<tr>
<td>Very High</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High to Moderate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Low</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Want to exploit all available P, T headroom
## Composing the Techniques

<table>
<thead>
<tr>
<th>Multicore Loading Condition</th>
<th>Bounding Constraints</th>
<th>Gain from combining?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBo(+Mig)</td>
<td>Paceline</td>
</tr>
<tr>
<td>T/P V PE T/P V/P_E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very High</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High to Moderate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Low</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Want to exploit all available P, T headroom
## Composing the Techniques

<table>
<thead>
<tr>
<th>Multicore Loading Condition</th>
<th>Bounding Constraints</th>
<th>Gain from combining?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBo(+Mig)</td>
<td>Paceline</td>
</tr>
<tr>
<td></td>
<td>T/P</td>
<td>V</td>
</tr>
<tr>
<td>High to Moderate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Low</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Common Case**

Want to exploit all available P, T headroom
## Composing the Techniques

| Multicore Loading Condition | Bounding Constraints | Gain from combining?
|-----------------------------|----------------------|-------------------------
|                             | VBo(+Mig) | Paceline | VBo+PI |
| T/P | V | T/P | P<sub>E</sub> | T/P | V/P<sub>E</sub> |
| Very High | ✓ | ✓ | ✓ | ✓ | Unlikely |
| High to Moderate | ✓ | ✓ | ✓ | ✓ | Likely |
| Low | ✓ | ✓ | ✓ | ✓ | Definitely |

Want to exploit all available P, T headroom

---

Ulya Karpuzcu

LeadOut 27
Composing the Techniques

<table>
<thead>
<tr>
<th>Multicore Loading Condition</th>
<th>Bounding Constraints</th>
<th>Gain from combining?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBo(+Mig)</td>
<td>Paceline</td>
</tr>
<tr>
<td></td>
<td>T/P V</td>
<td>T/P P_E</td>
</tr>
<tr>
<td>Very High</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
</tbody>
</table>

Additional techniques can be applied

Want to exploit all available P, T headroom
V/f Boosting + Paceline
V/f Boosting + Paceline

Boost core frequency $f$ beyond nominal limits by increasing $V$; tolerating occasional timing errors.
V/f Boosting + Paceline

Boost core frequency $f$ beyond nominal limits by increasing $V$; tolerating occasional timing errors.
**V/f Boosting + Paceline**

Boost core frequency $f$ beyond nominal limits by **increasing** $V$; tolerating occasional timing errors.

![Graph showing performance and frequency relationship](image)

- **Rated**
V/f Boosting + Paceline

Boost core frequency \( f \) beyond nominal limits by **increasing** \( V \); tolerating occasional timing errors.
V/f Boosting + Paceline

Boost core frequency $f$ beyond nominal limits by increasing $V$; tolerating occasional timing errors.
A Highly-Configurable CMP
A Highly-Configurable CMP

- Combining configurable TS with V/f Boosting...
A Highly-Configurable CMP

- Combining configurable TS with V/f Boosting...
- Base for configurable TS: Paceline
A Highly-Configurable CMP

• Combining configurable TS with V/f Boosting...
• Base for configurable TS: Paceline
• Base for V/f Boosting: More advanced than Turbo Boost
A Highly-Configurable CMP

- Combining configurable TS with V/f Boosting...
- Base for configurable TS: Paceline
- Base for V/f Boosting: More advanced than Turbo Boost
- Each core can independently change V/f
A Highly-Configurable CMP

- Combining configurable TS with V/f Boosting...
- Base for configurable TS: Paceline
- Base for V/f Boosting: More advanced than Turbo Boost
- Each core can independently change V/f
- The critical thread is moved among two available cores to avoid hot spot formation
A Highly-Configurable CMP

Interconnect

C1 C2

C3 C4

C5 C6

C7 C8

C9 C10

C11 C12

C13 C14

C15 C16
A Highly-Configurable CMP

Hardware Support

C1 C2 C9 C10

C3 C4

C5 C6

C7 C8

Interconnect

C11 C12

C13 C14

C15 C16

$ $ $ $ $
A Highly-Configurable CMP

Hardware Support

- Per-core V, f domains
A Highly-Configurable CMP

Hardware Support
- Per-core V, f domains
- Paceline in every pair
A Highly-Configurable CMP

**Hardware Support**
- Per-core V, f domains
- Paceline in every pair
- Per-core T sensors
A Highly-Configurable CMP

Hardware Support
• Per-core V, f domains
• Paceline in every pair
• Per-core T sensors

Constraints

Interconnect
A Highly-Configurable CMP

Hardware Support

- Per-core V, f domains
- Paceline in every pair
- Per-core T sensors

Constraints

- Per-core Voltage < $V_{\text{max}}$
A Highly-Configurable CMP

Hardware Support
- Per-core V, f domains
- Paceline in every pair
- Per-core T sensors

Constraints
- Per-core Voltage < $V_{max}$
- Hottest spot on die < $T_{max}$
A Highly-Configurable CMP

Hardware Support

- Per-core V, f domains
- Paceline in every pair
- Per-core T sensors

Constraints

- Per-core Voltage < $V_{\text{max}}$
- Hottest spot on die < $T_{\text{max}}$
- Per-core power < $P_{\text{max}}$
Modes of Operation

Unoptimized

Baseline per-thread power and performance
# Modes of Operation

V/f Boost

<table>
<thead>
<tr>
<th>Mode</th>
<th>Migration</th>
<th>$V &gt; V_{\text{NOM}}$</th>
<th># Cores per Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>V/f Boost</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
</tr>
</tbody>
</table>
# Modes of Operation

## V/f Boost + Migration

<table>
<thead>
<tr>
<th>Mode</th>
<th>Migration</th>
<th>$V &gt; V_{NOM}$</th>
<th># Cores per Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>V/f Boost</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
</tr>
<tr>
<td>V/f Boost + Migration</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
</tbody>
</table>
## Modes of Operation

### Paceline

<table>
<thead>
<tr>
<th>Mode</th>
<th>Migration</th>
<th>$V &gt; V_{NOM}$</th>
<th># Cores per Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>V/f Boost</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
</tr>
<tr>
<td>V/f Boost + Migration</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
<tr>
<td>Paceline</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
</tbody>
</table>
## Modes of Operation

### V/f Boost + Paceline

<table>
<thead>
<tr>
<th>Mode</th>
<th>Migration</th>
<th>( V &gt; V_{\text{NOM}} )</th>
<th># Cores per Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>V/f Boost</td>
<td></td>
<td>✓</td>
<td>1</td>
</tr>
<tr>
<td>V/f Boost + Migration</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
<tr>
<td>Paceline</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
<tr>
<td>V/f Boost + Paceline</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
</tr>
</tbody>
</table>
Optimization Problem
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads

• Use configurability to adapt to workload demands
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads

• Use configurability to adapt to workload demands
  • Speed up sequential sections of parallel apps
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads
• Use configurability to adapt to workload demands
  • Speed up sequential sections of parallel apps
  • Speed up interactive apps in multiprogram mix
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads
• Use configurability to adapt to workload demands
  • Speed up sequential sections of parallel apps
  • Speed up interactive apps in multiprogram mix
• R threads: Power-efficient baseline $V, f$
Optimization Problem

• At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads
• Use configurability to adapt to workload demands
  • Speed up sequential sections of parallel apps
  • Speed up interactive apps in multiprogram mix
• R threads: Power-efficient baseline V, f
• S threads: Increase V, f for max performance
Optimization Problem

- At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads
- Use configurability to adapt to workload demands

**Performance of speed-critical threads depends on (S,R)**

- Speed up interactive apps in multiprogram mix
- R threads: Power-efficient baseline V, f
- S threads: Increase V, f for max performance
Optimization Problem

- At any given time, CMP executes a mix of speed-critical (S) and throughput-oriented (R) threads.
- Use configurability to adapt to workload demands.

Performance of speed-critical threads depends on (S,R)

Problem:
1. Which mode to use for a speed-critical thread?
2. How to optimally set V/f for chosen mode?
A Practical Controller...

Goal: Maximize frequency of speed-critical threads; where throughput threads run at nominal V/f
A Practical Controller...

Goal: Maximize frequency of speed-critical threads; where throughput threads run at nominal V/f

- Global Controller sets execution mode for speed-critical threads (same mode for all)
A Practical Controller...

Goal: Maximize frequency of speed-critical threads; where throughput threads run at nominal V/f

- Global Controller sets execution mode for speed-critical threads (same mode for all)
A Practical Controller...

Goal: Maximize frequency of speed-critical threads; where throughput threads run at nominal V/f

- Global Controller sets execution mode for speed-critical threads (same mode for all)
- Independent Thread Controllers greedily optimize V/f for each speed-critical thread
A Practical Controller...

Goal: Maximize frequency of speed-critical threads; where throughput threads run at nominal V/f

- Global Controller sets execution mode for speed-critical threads (same mode for all)
- Independent Thread Controllers greedily optimize V/f for each speed-critical thread
Global Controller

Diagram:

- Global Controller
  - Mode
    - Thread Controller
      - f,V
      - Core
    - Mode
      - Thread Controller
      - f,V
      - Core
Global Controller

- Can devote two cores to each speed-critical thread?
Global Controller

- Can devote two cores to each speed-critical thread?
- Yes: Apply VBo+Pl to all speed-critical threads
Global Controller

- Can devote two cores to each speed-critical thread?
  - Yes: Apply VBo+Pl to all speed-critical threads
  - No: Apply VBoost to all speed-critical threads
Global Controller

- Can devote two cores to each speed-critical thread?
  - Yes: Apply VBo+Pl to all speed-critical threads
  - No: Apply VBoost to all speed-critical threads
- Heuristic is suboptimal only in rare T - limited cases
LeadOut Evaluation
LeadOut Evaluation

- Detailed 32nm leakage, temperature modeling
LeadOut Evaluation

- Detailed 32nm leakage, temperature modeling
- VARIUS [TSM08] process variation model
LeadOut Evaluation

- Detailed 32nm leakage, temperature modeling
- VARIUS [TSM08] process variation model
- Simulated LeadOut controller on 16-core CMP
LeadOut Evaluation

- Detailed 32nm leakage, temperature modeling
- VARIUS [TSM08] process variation model
- Simulated LeadOut controller on 16-core CMP
  - SPECint2000 benchmarks
LeadOut Evaluation

- Detailed 32nm leakage, temperature modeling
- VARIUS [TSM08] process variation model
- Simulated LeadOut controller on 16-core CMP
  - SPECint2000 benchmarks
  - 50 Monte Carlo die samples
Power-Performance Tradeoff

- Per Core Power (W)
- Performance

Increasing Load

Unoptimized Paceline

V/f-Boost

V/f-Boost + Paceline
Power-Performance Tradeoff

Per Core Power (W)

Performance

Unoptimized

Increasing Load

Unoptimized

Increasing Load

1 1.1 1.2 1.5 1.4

Performance

4 6 8 10 12 14
Power-Performance Tradeoff

Per Core Power (W) vs. Performance

- Unoptimized
- Paceline

Increasing Load

Performance vs. Power-Performance Tradeoff

Values:
- X-axis: Performance
- Y-axis: Per Core Power (W)

Legend:
- Unoptimized
- Paceline

Graph illustrates the tradeoff between per core power and performance for different load conditions.
Power-Performance Tradeoff

Per Core Power (W) vs. Performance

- **Unoptimized**
- **V/f-Boost**
- **Paceline**

Increasing Load
Power-Performance Tradeoff

- Per Core Power (W)
- Performance

- Unoptimized
- Paceline
- V/f-Boost
- V/f-Boost + Migration

Increasing Load
Power-Performance Tradeoff

Per Core Power (W)

Performance

- V/f-Boost + Paceline
- V/f-Boost + Migration
- Paceline
- V/f-Boost
- Unoptimized

Increasing Load

Ulya Karpuzcu

LeadOut
Multiple Speed-Critical Threads
Multiple Speed-Critical Threads

- As multicores scale to more cores, they will run a mixed workload
Multiple Speed-Critical Threads

• As multicores scale to more cores, they will run a mixed workload

• Leaves some cores idle
Multiple Speed-Critical Threads

• As multicores scale to more cores, they will run a mixed workload
  • Leaves some cores idle
  • Uses some set of cores for throughput (R)
Multiple Speed-Critical Threads

- As multicores scale to more cores, they will run a mixed workload
- Leaves some cores idle
- Uses some set of cores for throughput (R)
- Demands max sequential performance from a few latency sensitive threads (S)
Multiple Speed-Critical Threads

- As multicores scale to more cores, they will run a mixed workload
  - Leaves some cores idle
  - Uses some set of cores for throughput (R)
  - Demands max sequential performance from a few latency sensitive threads (S)
- Multi-programmed workload or application with limited parallelism
Multiple Speed-Critical Threads

- As multicores scale to more cores, they will run a mixed workload
  - Leaves some cores idle
  - Uses some set of cores for throughput (R)
  - Demands max sequential performance from a few latency sensitive threads (S)

R and S dynamically vary over time and from application to application
V/f Boosting

# Unoptimized Threads (R) vs. # Optimized Threads (S)
V/f Boosting

Not Feasible: $R + S > 16$
V/f Boosting

Not Feasible: $R + S > 16$
V/f Boosting

Not Feasible: $R + S > 16$

Up to 20% speed-up
V/f Boosting with Migration

# Optimized Threads (S)

# Unoptimized Threads (R)
V/f Boosting with Migration

Not Feasible: \( R + 2S > 16 \)
V/f Boosting with Migration

Not Feasible:
\[ R + 2S > 16 \]
V/f Boosting with Migration

Not Feasible:
\[ R + 2S > 16 \]

20% speed-up
Migration results in homogeneity
Paceline

Not Feasible: $R + 2S > 16$
Paceline

Not Feasible: \( R + 2S > 16 \)
Not Feasible: \( R + 2S > 16 \)

13% speed-up
V/f Boosting + Paceline

Not Feasible:
$R + 2S > 16$
Not Feasible: $R + 2S > 16$
V/f Boosting + Paceline

Not Feasible: R + 2S > 16

Up to 36% speed-up
Paceline: 13%
V/f Boosting: 20%
Global Controller...

Not Feasible: $R + 2S > 16$
Global Controller...

Not Feasible: 
R + 2S > 16

V/f Boosting + Paceline up to S = 8
Global Controller...

\[ R + 2S > 16 \text{ Not Feasible:} \]

Unoptimized Threads (R)

Optimized Threads (S)

V/f Boosting + Paceline up to S = 8
Global Controller...

Not Feasible:
\[ R + S > 16 \]

V/f Boosting
for \( S > 8 \)

V/f Boosting + Paceline
up to \( S = 8 \)
Conclusion

• Shut-down idle cores to run performance critical threads at higher than nominal frequencies
• V/f Boosting and Timing Speculation
• Individual application suboptimal

• LeadOut: A highly-configurable CMP
• Combining V/f Boosting and TS synergistically to unlock more performance
• 34% speedup at 220% power increase if half of the cores busy
LeadOut: Composing Low-Overhead Frequency-Enhancing Techniques for Single-Thread Performance in Configurable Multicores

Brian Greskamp, Ulya Karpuzcu, Josep Torrellas

http://iacoma.cs.uiuc.edu/
Constraint Prevalence

The numbers above the bars show the average power consumption of the enhanced thread on the slower core in the pair. This makes it slightly slower than one of both cores in the pair to be that of individual system.

To explain these performance trends, Figure 7.2 shows the limiting constraints for each technique and loading level. The size of each bar segment represents the fraction of samples from the three regimes of Table 4.1 appear in the figure as follows:

- When none of the cores are available (0)
- When only one core is available (1)
- When two cores are available (2)
- When four cores are available (4)
- When eight cores are available (8)
- When sixteen cores are available (16)

Finally, the slower core in the pair can be limited by either of the three regimes:

- Unoptimized
- VBoost
- VBo+Mig

The three regimes are as follows:

1. Performance limited by Emax
2. Performance limited by P
3. Performance limited by Vmax

<table>
<thead>
<tr>
<th>Limiting Constraint (%)</th>
<th># Available Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBoost</td>
<td>1</td>
</tr>
<tr>
<td>VBo+Mig</td>
<td>2</td>
</tr>
<tr>
<td>Paceline</td>
<td>4</td>
</tr>
<tr>
<td>VBo+Pl</td>
<td>8</td>
</tr>
<tr>
<td>Unoptimized</td>
<td>16</td>
</tr>
</tbody>
</table>

The number above each bar is the average power consumption of the enhanced thread under that configuration.

The constraint implementation: for simplicity, we set the assumption of an oracular experiments in which the particular constraint was the pseudo constraint. The improvement reaches 387% when all 16 cores are available.

The three regimes of Table 4.1 appear in the figure as follows:

- When none of the cores are available (0)
- When only one core is available (1)
- When two cores are available (2)
- When four cores are available (4)
- When eight cores are available (8)
- When sixteen cores are available (16)

Finally, the slower core in the pair can be limited by either of the three regimes:

- Unoptimized
- VBoost
- VBo+Mig

The numbers above the bars show the average power consumption of the enhanced thread on the slower core in the pair. This makes it slightly slower than one of both cores in the pair to be that of individual system.

To explain these performance trends, Figure 7.2 shows the limiting constraints for each technique and loading level. The size of each bar segment represents the fraction of samples from the three regimes of Table 4.1 appear in the figure as follows:

- When none of the cores are available (0)
- When only one core is available (1)
- When two cores are available (2)
- When four cores are available (4)
- When eight cores are available (8)
- When sixteen cores are available (16)

Finally, the slower core in the pair can be limited by either of the three regimes:

- Unoptimized
- VBoost
- VBo+Mig

The three regimes are as follows:

1. Performance limited by Emax
2. Performance limited by P
3. Performance limited by Vmax

<table>
<thead>
<tr>
<th>Limiting Constraint (%)</th>
<th># Available Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBoost</td>
<td>1</td>
</tr>
<tr>
<td>VBo+Mig</td>
<td>2</td>
</tr>
<tr>
<td>Paceline</td>
<td>4</td>
</tr>
<tr>
<td>VBo+Pl</td>
<td>8</td>
</tr>
<tr>
<td>Unoptimized</td>
<td>16</td>
</tr>
</tbody>
</table>

The number above each bar is the average power consumption of the enhanced thread under that configuration.

The constraint implementation: for simplicity, we set the assumption of an oracular experiments in which the particular constraint was the pseudo constraint. The improvement reaches 387% when all 16 cores are available.
LeadOut Performance

Assume: One speed-critical thread, several idle cores

- VBoost
- VBo+Mig
- Paceline
- VBo+Pl

Speedup (%) vs. # Available Cores
Thread Controller

- Greedily increase Leader f while P, T not at limit
- Set Leader V to meet $P_E$ target
- Treat Leader-Checker as GALS system (Attack-Decay control)
  - Throttle Checker f to keep coupling queue < 1/2 full
  - Set Checker V to guarantee error-free operation
Thread Controller

- Greedily increase Leader f while P, T not at limit
- Set Leader V to meet $P_E$ target
- Treat Leader-Checker as GALS system (Attack-Decay control)
  - Throttle Checker f to keep coupling queue $< 1/2$ full
  - Set Checker V to guarantee error-free operation

Every 1ms
Example TS Architectures

**Razor** [Ernst03]
Configurability: always-on
Checking granularity: stage-level
Functional correctness: correct
Example TS Architectures

**Razor** [Ernst03]
Configurability: always-on
Checking granularity: stage-level
Functional correctness: correct

**Paceline** [Greskamp07]
Configurability: on-demand
Checking granularity: at-retirement
Functional correctness: correct
LeadOut: Composing Low-Overhead Frequency-Enhancing Techniques for Single-Thread Performance in Configurable Multicores

Brian Greskamp, Ulya Karpuzcu, Josep Torrellas

http://iacoma.cs.uiuc.edu/
Outline

- CMP Design Goals
- Techniques
- Timing Speculation
- V/f Boosting
- LeadOut
- Optimization Problem
- Evaluation
Voltage-Frequency Boosting

Rajesh Kumar and Pat Gelsinger, Fall 2008 IDF