A Framework for Dynamic Energy Efficiency and Temperature Management (DEETM)

Michael Huang, Jose Renau, Seung-Moon Yoo, Josep Torrellas
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu/flexram
Motivation

- Increasingly high power consumption
  - High temperature
  - Inefficient use of energy

- Limitations of existing approaches
  - Static optimizations
  - Coarse grain dynamic management (DPM)
  - Inefficient temperature control (sleep)
  - Independent targets: temp, energy efficiency
Goal

Unified framework for Dynamic Energy Efficiency and Temperature Management (DEETM)

- **Temperature**: enforce limit while minimizing slowdown
- **Energy efficiency**: maximize energy saving while exploiting performance slack
Contribution: DTEEM Framework

- Existing limitations:
  - static application
  - coarse grain dynamic
  - inefficient techniques
  - independent targets:
    - temperature control
    - energy efficiency

- DEETM approach:
  - multiple techniques
  - dynamic
  - fine grain
  - order techniques for maximum efficiency
  - unified target
DEETM Framework

- Monitors temperature & execution slack

- Runs decision algorithm periodically: 
  \{Thermal, Slack\} components

- Activates low-power techniques
  - dynamically
  - incrementally
  - in prioritized order
Techniques

- Instruction filter cache
- Data cache subbanking
- Voltage scaling
- Voltage scaling: DRAM only
- Light sleep
Instruction Filter Cache

High power mode:
Time: 1
Energy: E

Low power mode:
Time: 1 + mr*1
Energy: e + mr*E
Techniques

- Instruction filter cache
- Data cache subbanking
- Voltage scaling
- Voltage scaling: DRAM only
- Light sleep
Chip Environment

- Processor-in-memory
- 64 lean processors – 2-issue static
- Optimized memory hierarchy
- Integrated thermal sensors and instruction counter
Individual Techniques-E*D

Normalized Energy-Delay Product

Normalized Average Power

Graph showing the relationship between normalized energy-delay product and normalized average power for different techniques:
- Sleep
- MemVolt
- VoltFreq
- SubBank
- IFilter

Micro’33, Dec 2000
Combinations - E*D
Summary

- Techniques ordered by efficiency (same order apply to both targets)

- System applies techniques in order, dynamically and incrementally
Thermal Algorithm

Macrocycle

Temp > MaxTemp

Yes → Add the next most efficient technique

No

Temp < MinTemp

Yes → Remove the last technique
Temperature Control - Limit

\[ Power_i^* = 75\% Power_i + 25\% Power_{i-1}^* \]
Temperature Control - Slowdown

![Graph showing temperature control slowdown](image-url)
Slack Algorithm

Test baseline IPC

Add technique; Test new effective IPC*

Slowdown enough?

Yes

Adjust duty cycle of technique if necessary

Every Macrocycle (HW/OS)

No

The First Few Microcycles (HW)

Effective IPC*: frequency-adjusted

Macrocycle

Microcycle
Slack - Energy Consumption

![Graph showing energy consumption vs slack with two lines: one labeled $E*D*D = \text{Constant}$ and another labeled Gradual.](image)

- $E*D*D = \text{Constant}$
- Gradual
Slack Misprediction

Used Slack/Slack

- Gradual
- Perfect

Slack (%)
Related Issues

- Algorithm interaction
- Selecting macrocycle
- Handling *Thermal Crisis* situation
- Reducing technique activation overhead
- Flexible technique ordering
- Hardware vs. software implementation
Conclusions

- **Effective & efficient temperature control**
  - very few macrocycles still over limit
  - 27% longer execution vs. 98% (by sleeping)

- **Efficient & accurate fine-grain exploitation of execution slack**
  - 5% slack $\Rightarrow$ 27% energy saved
  - small slack misprediction