Scheduler-Based DRAM Energy Management

V. De La Luz
A. Sivasubramaniam, M. Kandemir,
N. Vijaykrishnan and M.J. Irwin.

The Pennsylvania State University
Outline

- Introduction.
- DRAM memory architecture and power modes.
- Previous work.
- Scheduler-based energy management.
- Experimental setup.
- Results.
- Conclusions.
Introduction

- Energy consumption important for mobile and high-end computing systems.
- DRAMs have more power control capabilities.
- Previous work focused on hardware-based and compiler-based techniques.
- We propose an Scheduler-based technique to transition memory banks to low power modes.
DRAM Architecture

To/From CPU

Configuration Registers

Memory Controller

Module

Bank
Memory Operating Modes

- **Active**: 3.75 nJ, 30 cycles
- **Standby**: 0.83 nJ, 2 cycles
- **Napping**: 0.32 nJ, 30 cycles
- **Power-down**: 0.005 nJ, 9000 cycles
**Previous Work.**

- Inter-access times predictors. Delaluz, et. al. 
  
  [HPCA '01]

  - **Constant Threshold Predictor (CTP).**
    - The bank is gradually transitioned to lower power modes.
    - A bank that is not used in the past will not be used in the future.

  - **History Based Predictor (HBP).**
    - Estimates the next access time based on the previous access times.
    - Savings close to ideal.
Previous Work (2)

- Software-based approaches (compiler transformations and compiler-guided power mode transitions)
- Power management for peripherals (disks and network cards). Liu, et.al [ISLPED '00].
- Power Aware Page Allocation. Lebeck, et.al. [ASPLOS '00]
OS-based Energy Management

- OS has global view of the system.
- Information about actual physical frame allocation.
- The OS can determine points during execution of an application where banks would remain idle, so they can be transitioned to low power modes.
Scheduler-based Approach

- Transitions the banks at context switch time.
- We expect the same banks will be used in the next quantum.
- We need to keep track of the banks that are used by the application per time quantum.
- In order to do that, we propose the creation of a Bank Usage Table (BUT).
Bank Usage Table (BUT)

One row per application/task.

An X means a bank was used in the previous quantum.

<table>
<thead>
<tr>
<th>Process</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>...</th>
<th>Bn</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>X</td>
<td></td>
<td>X</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>...</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Pm</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
How do we update the BUT?

- **Page Permissions.** Wipe out access permissions on the page table for those virtual pages mapped to physical memory.

- **Page Reference Bit.** Reset reference bits on the page table.
- TLB Miss Handler. Modify the TLB miss handler. This is a hand-optimized, very efficient code. Not a good idea to modify it.

- Dump the TLB contents. Is expected that the TLB will do a good job capturing the program’s behavior.

We use the first approach in this work.
## Bank Usage Table (4)

<table>
<thead>
<tr>
<th>Permissions wiped out</th>
<th>$P_i$ executes</th>
<th>Other Processes Executed</th>
<th>Permissions wiped out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition idle</td>
<td>Page faults</td>
<td>Transition idle idle</td>
<td></td>
</tr>
<tr>
<td>memory banks</td>
<td>BUT is updated</td>
<td>memory banks</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 4: Bank Usage**

- **Permissions wiped out**: Indicates when permissions are reset.
- **Transition idle memory banks**: Tracks when processes transition to an idle state.
- **$P_i$ executes**: Shows execution of processes.
- **Page faults BUT is updated**: Displays page faults and updates.
- **Other Processes Executed**: Logs execution of other processes.

**Time**

---

---
Nothing prevents us from combining our scheduler-based technique with the hw-based ones (CTP, HBP).
Experimental Setup

- SUN UltraSparc 5, Linux RH 6.2.
- Kernel 2.2.14.
- 128 MB RAM, 16 banks.
- BUT rows implemented as part of the task structure in the kernel.
- Scheduler and page fault handler were modified.
- Complete, full-fledged operating system used for evaluation.
Experimental Setup (2)

- Traces
- Energy Models
- Energy Statistics
- P1 → ... → Pn
- Operating System
- Energy Estimation
- Memory Configuration
# Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>rawcaudio</td>
<td>MediaBench</td>
</tr>
<tr>
<td>rawdaudio</td>
<td>MediaBench</td>
</tr>
<tr>
<td>polyphase</td>
<td>MediaBench</td>
</tr>
<tr>
<td>md5</td>
<td>MediaBench</td>
</tr>
<tr>
<td>cordic</td>
<td>MediaBench</td>
</tr>
<tr>
<td>paraffins</td>
<td>Trimaran</td>
</tr>
<tr>
<td>g721encode</td>
<td>MediaBench</td>
</tr>
<tr>
<td>mcf</td>
<td>Spec2000</td>
</tr>
</tbody>
</table>
Multiple Instances of mcf

Normalized Energy

Number of Instances
Effect of Multiple Instances on Number of Banks (mcf)
Conclusions

- We presented an scheduler-based technique to transition memory banks to low power modes.
- We show that important savings can be achieved (up to 92%) with little HW support.
- Further improvement can be achieved by applying our technique along with existing hardware-based techniques.
Future Work

- Implementation of other mechanisms for updating the BUT.
- Comparison of energy savings among them.
- Study of other OS-based power-aware mechanisms.
Microsystems Design Laboratory

The Pennsylvania State University

Dept. of Computer Science and Engineering

http://www.cse.psu.edu/~mdl
Effect of Cache

Normalized Energy

- rawaudio
- rawaudio
- polyphase
- md5
- cordic
- paraffins
- g721encode
- mcf

HBP
HBP + BUT
HBP (cache)
HBP + BUT (cache)