Real-time Power Management for Energy Harvesting Embedded Multicore Systems

Hideki TAKASE, Kyungsoo LEE, Tohru ISHIHARA
Kyoto University
Advise

While we concentrate mainly on the methods available as an IC designer to reduce power, it should be remembered that it is the application and architectural level where the major impact on power dissipation can be made. ... To state this again, you must optimize power in a top-down manner, from the problem definition downward. Do not optimize from the bottom up, i.e., the circuit level; you will be doomed to fail.

by Neil H. E. Weste and David Money Harris
in 4th edition CMOS VLSI Design,
A Circuits and Systems Perspective
Research Target

- Computers embedded in social infrastructure
  - Integrated in human society and cooperated with humans
  - Both high-performance and high energy-efficiency are crucial
Agenda

• Efficiency
  - Generating efficiency
    ✓ Harvesting device efficiency
  - Consuming efficiency
    ✓ Consumer device efficiency
  - Transferring efficiency
    ✓ DC-DC converter efficiency
    ✓ Charger efficiency
    □ Harvested energy is not constant
    □ Power is not available on-demand
    □ High peak power is not available

1st part

2nd part
Background

- Low Power / Low Energy
  - Growth of Internet of Things (IoT) market

- Peak Performance
  - Growth of real-time applications

- Scalability and adaptability
  - Energy saving without losing the peak performance

- Real-time DVFS on a multi-core processor
Approaches to HPP & LAP

**DVFS processor**
- Voltage and frequency are dynamically changed
  - Widely used in GPP
- High design complexity and large transition overhead
- Prevent real-time operation

Not suitable for real-time embedded systems

**Heterogeneous exclusive multicore processor**
- Integrating HP-core and LP-core
- Exclusively running HP-core and LP-core
  - Fresh idea invented within the last 5 years

Not sufficiently studied
• Processor dynamically changes its clock frequency
• Programmers specify the clock frequency using a specific I/O instruction or API
• The operating voltage is adjusted to the minimum possible value which guarantees the correct operation of the processor
DVFS Pros and Cons

• Pros
  – Quadratic reduction of energy consumption

• Cons
  – Large overhead of DC-DC converter
  – Low performance at low operating voltage


This prevents real-time control of DVFS processors

\[ t_{TRAN} = \frac{2 \cdot C}{I_{MAX}} \cdot |V_{DD1} - V_{DD2}| \]

\[ E_{TRAN} = (1 - \eta) \cdot C \cdot |V_{DD1}^2 - V_{DD2}^2| \]

Typically 0.9

- C=100μF, I_{MAX}=1A
- V_{DD1}=1.0, V_{DD2}=0.68
- t_{TRAN}=64μs, E_{TRAN}=4.6μJ
# Commercial DVFS Processors

Large voltage transition overhead involved [6]

<table>
<thead>
<tr>
<th>Processor</th>
<th>Supply Voltage (V)</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmeta Crusoe</td>
<td>1.1V – 1.65V</td>
<td>300μs</td>
</tr>
<tr>
<td>AMD Mobile K6</td>
<td>0.9V – 2.0V</td>
<td>200μs</td>
</tr>
<tr>
<td>Intel PXA250</td>
<td>0.85V – 1.3V</td>
<td>500μs</td>
</tr>
<tr>
<td>Compaq Itsy</td>
<td>1.1V – 1.6V</td>
<td>189μs</td>
</tr>
<tr>
<td>TI TMS320C55x</td>
<td>1.1V – 1.6V</td>
<td>3.2ms (1.6V → 1.1V) 300μs (1.1V → 1.6V)</td>
</tr>
<tr>
<td>UCB[7]</td>
<td>1.2V – 3.8V</td>
<td>520μs</td>
</tr>
</tbody>
</table>

Issues in DVFS Processor Design

- The critical path changes depending on the $V_{DD}$
  - The circuit optimized for 1.0V may not be optimal for 0.7V
Heterogeneous Exclusive Multicore

- Integrating both HP and LP cores and exclusively activating them according to performance demands
  - Can reduce average power w/o losing the peak performance
  - Cores can be optimized for individual performance demands

- Commercially available examples
  - NVIDIA Variable SMP
    - NVIDIA Tegra3, Tegra4
  - ARM big.LITTLE
    - Samsung Exynos 5 Octa
Hetero-Exclusive Multicore (1/2)

- nVIDIA Variable SMP\(^2\)
- KnightShift\(^5\)

Hetero-Exclusive Multicore (2/2)

• **ARM big.LITTLE**[^3]

  ![Diagram showing ARM big.LITTLE architecture with A15 and A7 clusters for low and high load](image)

  ![Diagram showing a ODROID-XU board](image)

  ![Diagram showing an Arndale Octa Board](image)

• Samsung社 Exynos 5 Octa[^4]
  - ODROID-XU
  - Arndale Octa Board

Overhead of Task Migration

- Overhead involved in migrating tasks
  - Large time & energy overheads for migrating tasks
  - Many cache misses will occur after the task migration
  - Typical CPU employ tightly coupled cache and SPM
Multi-Performance Processor [1]

- MPP: Multi-Performance Processor
- Integration of multiple same-ISA CPU cores
  - Only one CPU core is activated at a time
- Pro: Each CPU core is optimized for its $f$ & $V_{DD}$
- Dynamically resizable cache (1-way~4-way)
- Con: Large area overhead involved

Prototype of MPP

- Designed based on MeP using 65nm CMOS process
- Fab provided cells are characterized for 1.2V and 0.7V
- HP and LP cores are optimized for 200MHz and 100MHz using 1.2V and 0.7V cell libs, respectively
Power Breakdown of MPP

Power of MPP when DCT is running

<table>
<thead>
<tr>
<th></th>
<th>200MHz Operation</th>
<th>100MHz Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way</td>
<td>34.3 mW</td>
<td>16.2 mW</td>
</tr>
<tr>
<td>1-way</td>
<td>16.2 mW</td>
<td>7.3 mW</td>
</tr>
</tbody>
</table>

- Leakage
- PLL
- SRAM
- Peripheral
- CPU core
- Measured

# cache ways
Change of CPU core

- 32 SPRs are transferred through dedicated bus
  - Only one delay cycle involved
- 28 GPRs and CRs are transferred through SPM
  - 15 General Purpose Registers
  - 13 Control Registers (Timer, interruption)
- Other registers are flushed along with core switch

![Diagram with MPP, Dedicated Bus, 100MHz 0.7V CPU core, 200MHz 1.2V CPU core, Stack (scratchpad memory), Time overhead 1.5μs, Energy overhead 10nJ, Small transition overhead, Making the finer-grained power management possible]
Traditional Approaches (1/2)

- Tasks are statically assigned to specific processors
  - No task migration across the processors
  - Cannot exploit intra-task performance variation
    - Energy efficiency of LP processor cannot be exploited

Pros: Easy to guarantee the real-time property
Cons: Energy reduction by load balancing is not possible
Traditional Approaches (2/2)

• Power management by lower-level drivers
  – Load balancing: Task migration across processors
  – If tasks are migrated w/o notifying it to OS ...
    ✓ Nobody knows the execution time of tasks

Pros: Can reduce energy by aggressive load balancing
Cons: Difficult to guarantee the real-time behavior
Power Management RTOS for MPP

- Required properties for a PM-RTOS running on MPP
  - Exploiting heterogeneity of MPP (Stay LP as long as possible)
  - Guaranteeing real-time processing for running tasks
    - Worst case execution time should be easily analyzed
    - Overhead of core switch and task migration should be small

OS should specify and memorize the execution modes of running tasks

- OS specifies exec. mode of CPU for every task instances
- Exploits extended Task Control Block (TCB) which memorize exec. modes of tasks
Implementation

- Execution modes of MPP
  - CPU core activated: High-speed CPU / Low-power CPU
  - Cache associativity value: way-1,2,3,4
    ✓ The execution mode is memorized in a specific register

- Implementation w/ TOPPERS next generation kernel
  - Extending the existing task management functionality
  - Providing power managing API

Improved real-time performance
Task Management Table

- Extending task control block (TCB)
  - config
  - p_tinib->iconfig
    ✓ Type is CFG (signed int)

- Extending the SPEC of CRE_TSK
  CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, PRI itskpri, SIZE stksz, STK_T *stk, CFG iconfig })

- Extension of functionality in task dispatcher
  - Every time a task is dispatched p_tcb->config is referenced
  - If the config needs to be updated, change the iconfig value
Power Management API

set_cfg : assigning the execution mode of CPU to the task

【API for C programming】
ER ercd = set_cfg(ID tskid, CFG config)

【Parameters】
ID tskid Task ID
CFG config Execution mode of CPU

【Returned parameters】
ER ercd (E_OK) if correctly done; otherwise error codes

【Error code】
E_CTX if called when CPU is locked
E_ID Illegal task ID number (tskid is incorrect)
E_PAR Illegal config number (config is incorrect)

※set_cor() and set_ica() are also implemented
Experimental Setup

• Measurement setup
  – Execution time: Gate-level simulation of entire MPP
  – Power: Measurement of $I$ and $V$ with Agilent U2722A
    ✓ Sampling rate: 60Hz
  – PM-RTOS: TOPPERS/ASP Kernel (Release 1.4.0)

• Measured items
  – Transition overheads for changing the MPP execution mode
  – Transition overheads for changing TCB in real-time OS
  – Average power consumption of running tasks on the RTOS
Overhead for MPP Mode Transition

- Transition time for the change of CPU
  - High-speed CPU → Low-power CPU: 1.65μs
  - Low-power CPU → High-speed CPU: 1.48μs
    ✓ Including store/resume of register values
    ✓ big.LITTLE architecture needs at least 20μs

- Change of cache associativity
  - High-speed mode: 25ns
  - Low-power mode: 50ns
    ✓ Corresponding 5 clock cycles

Very low overhead transition is possible

Not including the cache miss penalty caused by the change of cache associativity
Overhead in TOPPERS code

- Code size of TOPPERS sample1
  - Sample program for testing the basic functionality of the OS
  - # of tasks: 5 (MAINTASK, TASK1,2,3, LOGTASK)

<table>
<thead>
<tr>
<th>Power management</th>
<th>text</th>
<th>data</th>
<th>bss</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o PM</td>
<td>23,736</td>
<td>7,204</td>
<td>12,368</td>
<td>43,308</td>
</tr>
<tr>
<td>w/ PM</td>
<td>29,640</td>
<td>3,128</td>
<td>12,368</td>
<td>45,136</td>
</tr>
</tbody>
</table>

Code size is kept small
→ Suitable for the embedded applications

4.5%
Power Management Policy

- **Benchmark program**
  - 3 DCT tasks periodically invoked with different task periods
  - Rate-Monotonic scheduling (preemptive)

- **Power management policies**
  - Nonmanage: Always running with the highest speed
  - Static: Assigning execution modes of CPU to the tasks statically
  - Dynamic: Choose mode depending on the computational load
    - $\text{load} =$ the number of ready tasks
Power Consumption Results (1/2)

- 0.7V core (mW)
- 1.2V core (mW)
- 3.3V IO (mW)

Power Consumption [mW]

Nonmanage

Static

Dynamic
## Power Consumption Results (2/2)

- Evaluation results [mW]

<table>
<thead>
<tr>
<th>Policy</th>
<th>0.7V CPU</th>
<th>1.2V CPU</th>
<th>3.3V I/O</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonmanage</td>
<td>0.38</td>
<td>11.80</td>
<td>8.26</td>
<td>20.45</td>
</tr>
<tr>
<td>Static</td>
<td>0.58</td>
<td>9.50</td>
<td>5.80</td>
<td>15.88</td>
</tr>
<tr>
<td>Dynamic</td>
<td>0.73</td>
<td>7.47</td>
<td>2.66</td>
<td>10.87</td>
</tr>
</tbody>
</table>

The values include the power consumed by RTOS and that for mode transitions.

Using `set_cfg` API

- 23.3%
- 31.6%
Summary

• Summary
  – Implementation of multi-performance processor (MPP)
    ✓ Smaller mode-transition overhead than DVFS
    ✓ More energy efficient at low voltage than DVFS
  – Implementation of power management API for MPP
    ✓ Extending TCB of existing real-time OS
    ✓ Integrated on TOPPERS next generation kernel
    ✓ Real silicon validation of power management RTOS

• Future work
  – Optimizing basic cells, macro modules and architecture for low voltage operation
  – Developing more sophisticated PM algorithm
  – Developing PM-RTOS for multi processor
Energy Transferring Efficiency

- Efficiency
  - Generating efficiency
    - Harvesting device efficiency
  - Consuming efficiency
    - Consumer device efficiency
  - Transferring efficiency
    - DC-DC converter efficiency
    - Charger efficiency
    - Harvested energy is not constant
    - Power is not available on-demand
    - High peak power is not available
Power Loss in Portable System

Up to 15% of power is dissipated in DC-DC converters

Power Loss in Data Center

Power transferring loss when a computational load is 100W

Source: DC Power for Improved Data Center Efficiency, LBNL
Power Loss in DC-DC Conversion

- Power loss depends on
  - Boost-up or drop-down
  - Voltage difference
  - Output power

\[
\text{Power Loss} \propto (100 - \text{Efficiency}) \cdot P_{\text{output}}
\]

Motivations

- Many DC-DC converters are used in a system
  - Different components use different voltages
- Large power loss in the DC-DC converters
  - Voltage difference between source and load is large

Set the voltage here

Energy source

DC-DC
Supply voltage A group
- e.g.) CPU 1.0V

DC-DC
Supply voltage B group
- e.g.) modem 3.3V

DC-DC
Supply voltage C group
- e.g.) camera 5.0V
Reconfigurable Array

- An example for a configurable array
  - 3 configurations with 4 photovoltaic (PV) cells or super-capacitor cells
  - Each cell has 0.5V, 80mA output

- Reduce the difference between the input and output voltages in a DC-DC converter

(4,1): 0.5V, 320mA output

(2,2): 1V, 160mA output

(1,4): 2V, 80mA output


Proposed System Architecture

- System block diagram

Control the configuration to minimize the total loss

Power OR’ing by Linear Technology’s ideal diode
One Case Result for the Array Configuration

- Power loss in converters for different configurations

- CPU, modem, and camera consume 100mA, 30mA, and 1mA, respectively
- Sunlight intensity and state of charge in super-cap are 100% and 20% respectively

The lowest power loss configuration

Configuration \((m_c, n_c, m_{pv}, n_{pv})\)
Scheduling

• I/O aware task scheduling
  - Minimize the overlap among CPU tasks and I/O tasks
  - Make the chance of power reduction larger

Motivations

• Many DC-DC converters are used in a system
  ➢ Different components use different voltages
• Large power loss in the DC-DC converters
  ➢ Voltage difference between source and load is large

Set the voltage here

![Diagram showing the energy source and DC-DC converters for different components.](image)
Experimental Results

Conventional

Proposal

72% reduction in conversion loss
25% reduction in total power

Sunlight intensity
State of Charge of super-capacitor

Loss components:
- loss_conv
- loss_char
- loss_p
- loss_mod
- loss_cam

Proc.
Mod.
Cam.
Summary

- Dynamic selection of PV and Cap array configurations
  - Energy efficiency in each harvesting condition can be maximized

- Simultaneous array configuration and task scheduling
  - Multiple supply voltage loads are considered
  - Power loss in DC-DC converters can be reduced by 72%

- Contribution
  - System cost efficiency and quality of services can be improved
  - without oversizing a PV array or an energy storage
Conclusions

- Covered topics related to system and architectural-level design for energy efficient embedded systems
- High energy efficiency can be obtained by combined efforts at different design hierarchies
- Cross-layer optimization is an area of research with many opportunities
- Deep understanding of application and architectural-level is important for developing energy efficient circuit and its methodology